MAIARA APARECIDA COIMBRA VALENTIM

DECODIFICAÇÃO DE CÓDIGOS CORRETORES DE ERROS POR MEIO DE REDES NEURAIS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Matemática, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS - BRASIL
2019

Ficha catalográfica preparada pela Biblioteca Central da Universidade Federal de Viçosa - Câmpus Viçosa

T
V155d
2019

Valentim, Maiara Aparecida Coimbra, 1992-
Decodificação de códigos corretores de erros por meio de redes neurais / Maiara Aparecida Coimbra Valentim. – Viçosa, MG, 2019.
viii, 78 f. : il. (algumas color.) ; 29 cm.

Orientador: Marines Guerreiro.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f. 76-78.

1. Códigos corretores de erros (Teoria da informação). 2. Redes neurais (Computação). 3. Teoria dos grafos. I. Universidade Federal de Viçosa. Departamento de Matemática. Programa de Pós-Graduação em Matemática. II. Título.

CDD 22. ed. 519.7

MAIARA APARECIDA COIMBRA VALENTIM

DECODIFICAÇÃO DE CÓDIGOS CORRETORES DE ERROS POR MEIO DE REDES NEURAIS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Matemática, para obtenção do título de *Magister Scientiae*.

APROVADA: 17 de julho de 2019.

Sônia Maria Fernandes

Marcelo Firer

Marinês Guerreiro
(Orientadora)

Para Pe. Jorge e Irmã Catarina

A fé e a razão constituem como que
as duas asas pelas quais
o espírito humano se eleva
para a contemplação da verdade.

São João Paulo II

Agradecimentos

Agradeço a professora Marinês pela orientação, pela paciência, pelos conselhos e pelos ensinamentos valiosos que contribuíram muito para minha formação.

Agradeço aos professores, funcionários e colegas do DMA-UFV por todas as experiências compartilhadas. Em especial, agradeço aos amigos João e Ray por toda ajuda e por sempre terem tornado melhores os meus dias de estudo.

Agradeço também aos professores José Paulo, José Carlos e a professora Lucy pois, a confiança, o apoio, e o incentivo deles, foram fundamentais para que eu pudesse começar e continuar o mestrado.

Agradeço a CAPES pelo auxílio financeiro indispensável para a realização deste trabalho.

Agradeço a minha família e em especial a minha mãe, Fátima, por ser o melhor apoio, por ser alegria e um grande exemplo para mim. Agradeço ao meu noivo Tobias Fernando, que esteve comigo desde o começo desta caminhada, obrigada pelo incansável incentivo, pelo companheirismo e amor em todos os momentos.

Agradeço ainda aos amigos da UFJF e ao Ministério Universidades Renovadas de Juiz de Fora que foram neste período minha família e sustento. Além disto, agradeço de modo muito especial, ao Padre Jorge Luis Duarte e a Irmã Catarina a quem dedico este trabalho, por terem me acolhido e me ensinado tanto. Sem vocês, chegar até aqui não teria sido possível.

Assim, agradeço a Deus pelo dom da vida, pelos sonhos e bençãos, e por ter colocado cada uma destas pessoas em meu caminho.

Sumário

Resumo

VALENTIM, Maiara Aparecida Coimbra, M.Sc., Universidade Federal de Viçosa, julho de 2019. **Decodificação de códigos corretores de erros por meio de redes neurais**. Orientadora: Marinês Guerreiro.

Nesta dissertação estudamos principalmente métodos de decodificação de códigos corretores de erros lineares por meio de redes neurais apresentando algumas formas de relacionar tais conceitos. A partir do estudo de alguns tópicos da Teoria de Redes Neurais e da Teoria dos Grafos, fizemos uma comparação entre a decodificação por síndrome da Teoria Clássica dos Códigos Corretores de Erros e um algoritmo de decodificação utilizando redes neurais. Um dos principais resultados, envolvendo o modelo de Hopfield, prova que a decodificação de máxima verossimilhança em um código linear é equivalente a encontrar o máximo global de uma função de energia em uma rede neural. Os códigos lineares podem ser representados como estados estáveis das funções de energia. Assim, dado um código linear, uma rede neural pode ser construída de tal forma que cada máximo local na função de energia corresponda a uma palavra do código, e reciprocamente, cada palavra do código corresponda a um máximo local de uma função de energia.

Abstract

VALENTIM, Maiara Aparecida Coimbra, M.Sc., Universidade Federal de Viçosa, July, 2019. **Decoding error-correcting codes through neural networks**. Adiviser: Marinês Guerreiro.

The main goal of this work is to study methods of decoding linear error correction codes using neural networks by presenting some ways of relating such concepts. From the study of some topics of Neural Networks Theory and Graph Theory, we compared the syndrome decoding of error correcting codes in the Classical Theory with a decoding algorithm using neural networks. One of the main results, using the Hopfield model, proves that maximum likelihood decoding in a linear code is equivalent to finding the global maximum of an energy function in a neural network. Linear codes can be represented as stable states of the energy functions. Thus, given a linear code, a neural network can be constructed such that each local maximum in the energy function corresponds to one word of the code, and conversely, each word of the code corresponds to a local maximum of one energy function.

Introdução

Em 1948, o matemático, engenheiro eletrônico e criptógrafo Claude Elwood Shannon publicou o artigo "*Mathematical Theory of Communication*" [40] no qual declarava que "o problema fundamental da comunicação é reproduzir em um ponto exatamente ou aproximadamente uma mensagem selecionada em outro ponto." Esta publicação deu início a um ramo da matemática intitulado Teoria da Informação e ou Teoria dos Códigos Corretores de Erros.

Esta teoria envolve várias áreas do conhecimento, tais como Matemática, Ciência da Computação, Engenharia Elétrica, com diversas aplicações, e seu problema fundamental consiste em determinar qual mensagem foi enviada com base no que é recebido [22].

Dado um código, o processo de codificação é geralmente simples, entretanto, o processo de decodificação, isto é, determinar qual palavra do código (e, portanto, qual mensagem "a") foi enviada quando a mensagem "a' " é recebida, caracteriza-se como um processo mais complexo. Encontrar algoritmos de decodificação eficientes (rápidos) é uma área importante de pesquisa na Teoria de Códigos Corretores de Erros devido as suas aplicações práticas [22, p. 39].

Na busca de encontrar bons algoritmos de decodificação que sejam mais rápidos, mais precisos, com melhor desempenho da correção de erros, ou mais fáceis de serem implementados computacionalmente, relacionou-se o problema de decodificação à Teoria de *Redes Neurais Artificiais*, como pode ser visto, por exemplo, em [16] e [34].

As redes neurais artificiais, ou simplesmente redes neurais, como são usualmente denominadas, constituem uma vasta área de pesquisa que tem sido aplicada em diversas outras áreas de conhecimento e são consideradas muito úteis para resolver problemas complexos, que são difíceis de serem resolvidos utilizando técnicas convencionais conhecidas e bem desenvolvidas [12].

O trabalho em redes neurais artificiais foi motivado pelo reconhecimento de que o cérebro humano processa informações de uma forma inteiramente diferente de um computador digital convencional. Por exemplo, o cérebro realiza rotineiramente tarefas de reconhecimento perceptivo, como a de reconhecer uma face familiar incorporada em uma cena desconhecida, em aproximadamente 100 a 200 milissegundos. Por outro lado, tarefas de complexidade muito menor demoram muito mais tempo para serem executadas em um computador [17, p. 27].

Deste modo, as redes neurais artificiais são definidas como modelos computacionais inspirados no sistema nervoso de seres vivos [41]. São máquinas projetadas para modelar a maneira como o cérebro humano executa uma tarefa particular ou uma função de interesse e, geralmente, implementadas utilizando componentes eletrônicos ou simuladas em software em um computador digital [17]. Elas possuem a capacidade de aquisição e manutenção do conhecimento (baseado em informações) e podem ainda ser definidas como um conjunto de unidades de processamento, caracterizadas por neurônios artificiais (similares à estrutura

do cérebro humano), que são interligados por um grande número de interconexões, as *sinapses artificiais* [41].

Um dos modos de se utilizar as redes neurais na decodificação de códigos corretores de erros [6] é considerá-las como grafos orientados e, por meio de propriedades dos grafos, podemos descrever uma família de códigos intitulada *códigos teóricos de grafos*. Partindo disto, para resolver o problema de decodificação de máxima verossimilhança, ou seja, encontrar a palavra mais próxima possível do que foi transmitido, recorremos a resultados relacionados ao *modelo* e à chamada *função de energia* que foram definidos por John Hopfield, em 1982, e estendemos os resultados obtidos, nos códigos teóricos de grafos, para os códigos lineares, de acordo com [6].

O objetivo deste trabalho é estudar algoritmos de decodificação para códigos corretores de erros por meio de redes neurais artificiais apresentando algumas formas de relacionar tais conceitos. Para isto, estudaremos a Teoria de Códigos Corretores de Erros e os princípios necessários da Teoria de Redes Neurais Artificiais e da Teoria de Grafos que a ela se relaciona, exemplos de decodificação são feitos e propriedades importantes que unem tais conceitos são demonstradas, conforme [6], [16].

Este trabalho está organizado em quatro capítulos. No Capítulo 1 apresentamos os principais tópicos sobre a Teoria dos Códigos Corretores de Erros que serão utilizadas em todo o trabalho. Para auxiliar na compreensão de exemplos, são abordados, especificamente, os códigos de Hamming e Reed-Muller e os códigos encurtados.

No Capítulo 2, apresentamos alguns tópicos da Teoria de Redes Neurais e da Teoria de Grafos que se fazem necessários para a compreensão dos capítulos subsequentes. Estabelecemos uma relação entre um neurônio biológico e um neurônio artificial. Apresentamos alguns pontos importantes a cerca da arquitetura e aprendizagem das redes neurais e ainda um resumo histórico sobre o estudo em redes neurais. A Seção 2.2 relacionada a grafos apresenta definições e exemplos.

No Capítulo 3, estabelecemos a relação entre redes neurais e grafos. Introduzimos o modelo de Hopfield, sua propriedade de convergência juntamente com uma função de energia associada ao modelo. Definimos os códigos teóricos de grafos, apresentamos uma cota superior para sua distância mínima e estabelecemos a relação destes com as redes neurais. Após, estendemos os resultados obtidos para códigos teóricos de grafos, provando que encontrar a decodificação de máxima verossimilhança em um código linear é equivalente a encontrar o máximo global de uma função de energia em uma rede neural. Além disto, provamos que, dado um código linear, uma rede neural pode ser construída de tal forma que cada máximo local na função de energia corresponda a uma palavra do código e, reciprocamente, cada palavra do código corresponda à um máximo local de uma função de energia.

No Capítulo 4 é feita uma comparação com a decodificação por síndrome da Teoria Clássica dos Códigos Corretores de Erros e um algoritmo de decodificação utilizando redes neurais. Assim, por meio de uma rede neural adaptada para corrigir uma palavra do código que apresenta no máximo um erro, um decodificador de código é construído. Neste processo, cada camada da rede neural simula um estágio da decodificação por síndrome de um código linear. O decodificador exibido se mostra útil uma vez que facilita a implementação computacional do algoritmo em tempo real.

Capítulo 1

Tópicos sobre Códigos Corretores de Erros

No processo de transmissão de informação, o meio pelo qual as informações são enviadas é chamado de **canal.** Ao enviar uma mensagem x, se nenhuma modificação for feita e ela for transmitida diretamente pelo canal, qualquer **ruído**[1] pode distorcer a mensagem fazendo com que talvez ela possa não ser recuperável.

A idéia básica, ao transmitir uma informação é, assim, "embelezar" a mensagem, adicionando alguma redundância a ela, de forma que a mensagem recebida seja de fato a mensagem original que foi enviada.

Deste modo, um **código corretor de erros** é um modo organizado de acrescentar dados adicionais, redundâncias, a cada informação que se queira transmitir ou armazenar, que permita, ao recuperar a informação, detectar e corrigir erros [14]. A este processo de acrescentar dados adicionais denominamos **codificação** e, ao processo de recuperar a informação, denominamos **decodificação**.

Para definir um código corretor de erros, precisamos de um conjunto finito e não vazio $\mathcal{A}$ que chamamos de **alfabeto**.

O **número de elementos** de $\mathcal{A}$ é denotado por $|\mathcal{A}|$ e simbolizado por q. O conjunto das $n-$uplas com entradas em $\mathcal{A}$ é denotado por $\mathcal{A}^n = \mathcal{A} \times \mathcal{A} \times \cdots \times \mathcal{A}$. Definimos:

Definição 1.1 *Um* ***código corretor de erros q-ário*** C *é um subconjunto próprio de* $\mathcal{A}^n$, *para algum número* $n \in \mathbb{N}$.

Definição 1.2 *Os elementos de* C *são chamados de* ***palavras do código****. Uma palavra* v *de comprimento* n *escrita com o alfabeto* $\mathcal{A}$ *é uma sequência* $v = (a_1, a_2, \ldots, a_n,)$ *(ou* $v = a_1 a_2 \ldots a_n$*), com* $a_i \in \mathcal{A}$, *para todo* i.

Um código corretor de erros $q-$ário C pode ainda ser chamado de código de blocos. Durante o texto, quando não houver possibilidade de confusão, nos referiremos ao "código de blocos" simplesmente como "código".

[1] O ruído consiste numa alteração de alguma das características do sinal transmitido por efeito de um outro sinal exterior ao sistema de transmissão, ou gerado pelo próprio sistema. Assim, ele é dependente do meio, por exemplo, em um disco compacto o ruído pode ser causado por impressões digitais ou arranhões no disco, em dispositivos de comunicação sem fio o ruído pode ser causado por radiação eletromagnética, etc...

A transmissão de mensagens num sistema de comunicação acontece do seguinte modo. Uma mensagem, para ser transmitida, deve ser codificada, acrescentando redundâncias através de um **processo de codificação**. Após codificada, a mensagem é transmitida por um canal e, ao chegar ao remetente, deve ser decodificada. No processo de decodificação, o algoritmo utilizado deve ser capaz de detectar os possíveis erros na mensagem recebida e, caso contenha erros, deve ser capaz de corrigí-los para, enfim, reconhecer a palavra enviada.

A Figura 1.1 ilustra o processo de transmissão de informação.

Figura 1.1: Sistema de comunicação: "a" denota uma mensagem enviada, "x" a mensagem codificada, "y" a mensagem transmitida e "a' " a mensagem recebida.

Para este capítulo as principais referências são [18], [22], [28] e [37]. A maior parte dos resultados deste capítulo serão apenas enunciados. As demonstrações podem ser encontradas em [18].

No envio de uma mensagem, se a palavra recebida contiver algum erro busca-se a palavra do código mais próxima da palavra que foi transmitida. Para identificar tal palavra precisamos comparar a palavra recebida com as palavras do código. Este conceito de "comparação entre palavras" é o que definimos como "medir" a distância entre palavras em $\mathcal{A}^n$. Uma forma de medir esta distância é através da *Distância de Hamming:*

Definição 1.3 *Dados dois elementos* $u = (u_1, \ldots, u_n), v = (v_1, \ldots, v_n) \in \mathcal{A}^n$, *a* ***distância de Hamming*** *entre* u *e* v *é dada por*

$$d(u, v) = |\{i : u_i \neq v_i, 1 \leq i \leq n\}|.$$

Definição 1.4 *Seja* $C \subset \mathcal{A}^n$ *um código. A* ***distância mínima*** *de* $\mathcal{C}$ *é o número*

$$d = d(C) = min\{d(u, v) : u, v \in C \text{ e } u \neq v\}.$$

Por satisfazer as propriedades de *métrica* a distância de Hamming é também chamada de *métrica de Hamming*. O resultado abaixo caracteriza este fato:

Proposição 1.1 *[18, Proposição 1.1] A distância de Hamming determina uma métrica em* $\mathcal{A}^n$, *ou seja, dados* u, v *e* $w \in \mathcal{A}^n$, *valem as seguintes propriedades:*

- *Positividade:* $d(u, v) \geq 0$ *e vale a igualdade se, e somente se,* $u = v$.
- *Simetria:* $d(u, v) = d(v, u)$.
- *Desigualdade Triangular:* $d(u, v) \leq d(u, w) + d(w, v)$.

A Definição 1.5 e os Lemas 1.1 e 1.2 a seguir, fornecem uma interpretação diferente para a definição da distância mínima, uma forma também de determinar quando uma palavra pertence ou não ao código.

Definição 1.5 *Dados um elemento $a \in \mathcal{A}^n$ e $t \geq 0$ um número real, define-se* ***disco*** *e* ***esfera*** *de centro a e raio t, respectivamente, por:*

$$D(a;t) = \{u \in \mathcal{A}^n : d(u,a) < t\} \quad e \quad S(a;t) = \{u \in A^n : d(u,a) = t\}.$$

Lema 1.1 *[18, Lema 1.1] Para todo $a \in \mathcal{A}^n$ e todo número real $r > 0$, temos*

$$|D(a;r)| = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i.$$

Lema 1.2 *[18, Lema 2.1] Seja $\mathcal{C}$ um código com distância mínima d. Se c e c' são palavras distintas de $\mathcal{C}$, então*

$$D(c;\kappa) \bigcap D(c';\kappa) = \emptyset.$$

onde $\kappa = [\frac{d-1}{2}]$ e $[t]$ representa a parte inteira de um número real t.

Teorema 1.1 *[18, Teorema 1.1] Seja C um código q-ário de comprimento n com distância mínima d. Então C pode detectar até $d-1$ erros e corrigir até $\kappa = [\frac{d-1}{2}]$ erros.*

Demonstração: Suponha que ao transmitirmos uma palavra c do código cometemos t erros com $t \leq \kappa$, recebendo a palavra r, então $d(r,c) = t \leq \kappa$. Pelo Lema 1.2, a distância de r a qualquer outra palavra do código é maior do que κ. Isso determina c univocamente a partir de r, corrigindo a palavra recebida e substituindo-a por c.

Por outro lado, dada uma palavra do código, podemos nela introduzir até $d-1$ erros sem encontrar outra palavra do código e, assim, a detecção de erros será possível.

■

Note que, pelo Teorema 1.1, um código terá maior capacidade de correção de erros quanto maior for a sua distância mínima. O que exemplifica a importância de se obter ao menos uma cota inferior para a ela.

Definição 1.6 *Sejam $\mathcal{C} \subset \mathcal{A}^n$ um código q-ário com distância mínima d e $\kappa = [\frac{d-1}{2}]$. O código $\mathcal{C}$ será dito* ***perfeito*** *se*

$$\bigcup_{c \in C} D(c;\kappa) = \mathcal{A}^n.$$

Definição 1.7 *Um código C sobre um alfabeto $\mathcal{A}$ possui três* ***parâmetros*** *fundamentais $[n, M, d]$, que são, respectivamente, o seu comprimento (o número n correspondente ao espaço ambiente $\mathcal{A}^n$ em que C se encontra), o seu número de elementos M e a sua distância mínima d.*

1.1 Códigos Lineares

A classe de códigos mais utilizada na prática é a classe dos *códigos lineares*. Durante este trabalho será com esta classe de códigos que iremos trabalhar.

Para os códigos lineares, tomamos o alfabeto como um corpo finito $\mathbb{K}$ com q elementos e, para $n \in \mathbb{N}$, $\mathbb{K}^n$ é o espaço vetorial sobre $\mathbb{K}$ de dimensão n.

Definição 1.8 *Um código q-ário $C \subset \mathbb{K}^n$ é um* ***código linear*** *se for um* ***subespaço vetorial próprio*** *de $\mathbb{K}^n$.*

Os parâmetros de um código linear C são definidos como (n, k, d) com n o comprimento do código, k a dimensão do código sobre o corpo $\mathbb{K}$ e d sua distância mínima. Assim, C é um $(n, k)-$código linear, ou ainda, quando se conhece a distância mínima d de C, dizemos que C é um $(n, k, d)-$código linear. Sejam k a dimensão de um código C e $\{\mathbf{v_1}, ..., \mathbf{v}_k\}$ uma de suas bases. Todo elemento de C se escreve como combinação linear, de modo único, na forma

$$\lambda_1\mathbf{v_1} + \lambda_\mathbf{2}\mathbf{v_2} + ... + \lambda_k\mathbf{v}_k,$$

com $\lambda_i \in \mathbb{K}$, para todo $i = 1, ..., k$. Daí $M = |C| = q^k$ e, conseqüentemente, $dim_{\mathbb{K}}C = k = log_q q^k = log_q M$.

Definição 1.9 *Dado $x \in \mathbb{K}^n$, o* ***peso*** *de x é o número inteiro*

$$\omega(x) := |\{i : x_i \neq 0\}|.$$

Em outras palavras, $\omega(x) = d(x, 0)$, com d a métrica de Hamming.

Definição 1.10 *O* ***peso de um código*** *linear C é o inteiro*

$$\omega(C) := min\{\omega(x) : x \in C\backslash\{0\}\}.$$

Proposição 1.2 *[18, Proposição 2.5] Seja $C \subset \mathbb{K}^n$ um código linear com distância mínima d. Temos:*

(i) Para quaisquer $x, y \in \mathbb{K}^n$, $d(x, y) = \omega(x - y)$.

(ii) $d = \omega(C)$.

Pelo item (ii) da Proposição 1.2, a distância mínima de um código C também será chamada de peso do código.

Um subespaço vetorial C de um espaço vetorial $\mathbb{K}^n$ pode ser descrito como **imagem** ou como **núcleo** de uma transformação linear. Assim, sejam $\{e_1, \ldots, e_k\}$ a base canônica de $\mathbb{K}^k$ e $\{v_1, ..., v_k\}$ uma base de um código $C \subset \mathbb{K}^n$, para $k, n \in \mathbb{N}^*$, com $k < n$. Então C é isomorfo a $\mathbb{K}^k$. Podemos definir uma aplicação linear injetora $T : \mathbb{K}^k \longrightarrow \mathbb{K}^n$ por $T(e_i) = v_i$, para $0 \leq i \leq k$ e, por construção de T, temos $Im(T) = C$. Agora, para descrever C por uma transformação linear sobrejetora $T' : \mathbb{K}^n \longrightarrow \mathbb{K}^{n-k}$ tal que $Ker(T') = C$ podemos completar a base $\{v_1, ..., v_k\}$ de C a uma base $\{v_1, ..., v_k, c_1, ..., c_{n-k}\}$ de $\mathbb{K}^n$.

Assim, cada $v \in \mathbb{K}^n$ pode ser escrito como $v = \lambda_1 v_1 + \ldots + \lambda_k v_k + \lambda_{k+1} c_1 + \ldots + \lambda_n c_{n-k}$, com $\lambda_i \in \mathbb{K}, 1 \leq i \leq n$. Daí, basta definirmos $T' : \mathbb{K}^n \longrightarrow \mathbb{K}^{n-k}$ por $v \longmapsto v' = \lambda_{k+1} c_1 + \ldots + \lambda_n c_{n-k}$ e obtemos $Ker(T') = C$.

1.1.1 Matriz Geradora de um Código

Seja $\beta = \{v_1, ..., v_k\}$ uma base ordenada de C. Escrevendo os elementos da base β como combinação linear dos elementos da base canônica $\{u_1, ..., u_n\}$ de $\mathbb{K}^n$, obtemos

$v_i = b_{i1}u_1 + b_{i2}u_2 + \ldots + b_{in}u_n$ para certos $b_{ij} \in \mathbb{K}$, com $(i,j) \in \{1,\ldots,k\} \times \{1,\ldots,n\}$. Como $T(e_i) = v_i$, para $0 \leq i \leq k$, a matriz de T nas respectivas bases canônicas é

$$G = \begin{pmatrix} v_1 \\ \vdots \\ v_k \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ \vdots & \vdots & & \vdots \\ b_{k1} & b_{k2} & \cdots & b_{kn} \end{pmatrix}.$$

Como $Im(T) = C$, cada vetor-linha de G pertence ao código C, ou seja, C é o subespaço gerado pelas linhas de G, que formam uma base de C. Assim, os elementos de C são todas as palavras $y \in \mathbb{K}^n$ tais que $xG = y$, para $x \in \mathbb{K}^k$.

Definição 1.11 *Uma matriz $G \in M_{k\times n}(\mathbb{K})$ cujas linhas formam uma base para C diz-se uma* ***matriz de codificação*** *ou* ***matriz geradora*** *de C em relação às bases canônicas de $\mathbb{K}^k$ e $\mathbb{K}^n$.*

Definição 1.12 *Uma matriz geradora G de um código C está na* ***forma padrão*** *se $G = (Id_k : A)$, com Id_k a matriz identidade de ordem k e A uma matriz qualquer $k \times (n-k)$.*

Dado um código C, sempre é possível obtermos uma matriz geradora de C na forma padrão utilizando operações sobre linhas ou colunas e sempre existe um código C', equivalente a C, com matriz geradora na forma padrão.

1.1.2 Matriz Teste de Paridade

Uma outra forma de definirmos um código linear é a partir de uma matriz teste de paridade. Denotamos por $H = (h_{ij})_{i,j} \in M_{(n-k)\times n}(\mathbb{K})$ a matriz de posto $(n-k)$ que representa a transformação linear

$$\begin{array}{rccc} T': & \mathbb{K}^n & \longrightarrow & \mathbb{K}^{n-k} \\ & v & \longmapsto & v' = \lambda_{k+1}c_1 + \ldots + \lambda_n c_{n-k}, \end{array}$$

nas bases canônicas destes espaços. Como $Ker(T') = C$, o código linear C é o conjunto de todas palavras $x \in \mathbb{K}^n$ satisfazendo $Hx^t = 0$.

Definição 1.13 *A matriz H construída acima diz-se uma* ***matriz teste de paridade*** *do código linear C.*

1.1.3 Códigos Duais

Considerando as transformações lineares T e T' definidas nas Seções 1.1.1 e 1.1.2, note que, para $x \in \mathbb{K}^k$, $T' \circ T(x) = T'(T(x)) = 0$, pois $T(x) \in Im(T) = C = Ker(T')$. O que implica, em termos de matrizes, $G \cdot H = 0$.

Dados $u = (u_1,\ldots,u_n)$ e $v = (v_1,\ldots,v_n) \in \mathbb{K}^n$, o **produto interno** de u e v é dado por

$$\langle u, v\rangle = u_1v_1 + \cdots + u_nv_n,$$

em que $< \cdot, \cdot >$ é o produto interno canônico.

Definição 1.14 *Seja $C \subset \mathbb{K}^n$ um código linear, o* ***conjunto ortogonal a*** *C em $\mathbb{K}^n$ é o conjunto*

$$C^{\perp} = \{v \in \mathbb{K}^n : \langle u, v\rangle = 0, \text{para todo } u \in C\}.$$

Lema 1.3 *[18, Lema 1.5] Se $C \subset \mathbb{K}^n$ é um código linear, com matriz geradora G, então:*

i) $C^{\perp}$ é um subespaço vetorial próprio de $\mathbb{K}^n$.

ii) $C^{\perp} = \{x \in \mathbb{K}^n : \ G \cdot x^t = 0\}$, ou seja, $x \in C^{\perp}$ se, e somente se, $Gx^t = 0$.

Definição 1.15 *O subespaço vetorial $C^{\perp}$ de $\mathbb{K}^n$ é um código linear chamado de* ***código dual*** *de C.*

Proposição 1.3 *[18, Proposição 2.5] Seja $C \subset \mathbb{K}^n$ um código de dimensão $k < n$ com matriz geradora $G = (Id_k : A)$ na forma padrão. Então*

i) $\dim C^{\perp} = n - k$,

ii) $H = (-A^t | Id_{n-k})$ é uma matriz geradora de $C^{\perp}$ e

iii) $(C^{\perp})^{\perp} = C$.

Para identificar se uma sequência $v \in \mathbb{K}^n$ pertence ou não a um código $C \subset \mathbb{K}^n$ podemos utilizar a seguinte proposição:

Proposição 1.4 *[18, Proposição 4.5] Seja $C \subset \mathbb{K}^n$ um código linear tal que $C^{\perp}$ tem matriz geradora H. Então*

$$v \in C \iff Hv^t = 0.$$

Note que a matriz H geradora de $C^{\perp}$ é uma matriz de teste de paridade de C.

Definição 1.16 *Dados um código linear $C \subset \mathbb{K}^n$ com matriz teste de paridade H e um vetor $v \in \mathbb{K}^n$, dizemos que Hv^t é a* ***síndrome*** *de v.*

Pode-se obter pela matriz teste de paridade informações à respeito do peso d de um código:

Proposição 1.5 *[18, Proposição 5.5] Dado um código $C \subset \mathbb{K}^n$ com matriz teste de paridade H, o peso de C é maior ou igual a s se, e somente se, quaisquer $s-1$ colunas de H são linearmente independentes.*

Teorema 1.2 *[18, Teorema 2.5] Seja H a matriz teste de paridade de um código C. Então $\omega(C) = s$ se, e somente se, quaisquer $s-1$ colunas de H são linearmente independentes e existem s colunas de H linearmente dependentes.*

Corolário 1.1 *[18, Corolário do Teorema 2.5] (**Cota de Singleton**) Os parâmetros (n, k, d) de um código linear satisfazem à desigualdade*

$$d \leq n - k + 1.$$

Se em um código vale a igualdade $d = n-k+1$, tal código é chamado de **MDS** (*Maximum Distance Separable*).

1.1.4 Códigos de Hamming e Reed-Muller

Nesta seção descrevemos dois tipos de códigos, os **códigos de Hamming** e os **códigos de Reed-Muller**, que serão utilizados em exemplos nos capítulos seguintes.

Os códigos de Hamming foram criados por Richard Hamming, na mesma época em que Shannon desenvolvia seus estudos, diante da necessidade de correção de erros em seu trabalho em computadores. Tais códigos também marcaram o início da Teoria da Informação, sendo de grande importância tanto por razões teóricas e práticas quanto por razões históricas.

Os códigos Reed-Muller são também uma das famílias de códigos mais antigas e melhor compreendidas [28]. Foram descobertos em 1954 por Muller e o primeiro algoritmo de decodificação foi desenvolvido por Reed também em 1954.

Definição 1.17 *Um **código de Hamming** de ordem m sobre $\mathbb{F}_2$ é um código C com matriz teste de paridade H_m de ordem $m \times n$, cujas colunas são os elementos de $\mathbb{F}_2^m \backslash \{0\}$ numa ordem qualquer.*

Assim, seja $C \subset \mathbb{F}_2^n$ o código determinado pela matriz H_m, temos $n = 2^m - 1$, pela própria construção de H_m. Com isso, sua **dimensão** é $k = n - m = 2^m - m - 1$.

A **distância mínima** em um código de Hamming é $d = 3$, pois é fácil encontrar 3 colunas linearmente dependentes em H_m.

Exemplo 1.1 *Considerando a matriz*

$$H_3 = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}, \quad \begin{array}{l} m = 3 \\ n = 2^3 - 1 = 7 \\ k = 7 - 3 = 4 \end{array}$$

Proposição 1.6 *[18, Proposição 6.5] Todo código de Hamming é perfeito.*

Demonstração: Como $d = 3$, temos $\kappa = [\frac{d-1}{2}] = [\frac{3-1}{2}] = 1$. Dado $c \in \mathbb{F}_2^n$, temos

$$|D(c;1)| = n + 1.$$

Portanto, $|\bigcup_{c \in C} D(c;1)| = [n+1]2^k = [2^m - 1 + 1]2^{n-m} = 2^n$, logo

$$\bigcup_{c \in C} D(c;1) = \mathbb{F}_2^n \quad \text{e o código é perfeito.}$$

■

Definição 1.18 *Para cada inteiro positivo m e para cada r inteiro não negativo tais que $r \leq m$, podemos construir um código binário de comprimento 2^m. Para cada comprimento haverá $m+1$ códigos lineares, denotados por $R(r,m)$ e chamados códigos de **Reed-Muller** de $r-$ésima ordem e de comprimento 2^m.*

Os códigos $R(0,m)$ e $R(m,m)$ são chamados códigos de Reed Muller triviais. Para $m \geq 0$, $R(0,m)$ é o código binário de repetição de comprimento 2^m, com base $\{1\}$, e $R(m,m)$, para $m \geq 1$, é o espaço inteiro $\mathbb{F}_2^{2^m}$. Por exemplo:

$$R(0,0) = \{0,1\} = \mathbb{F}_2, \quad R(0,1) = \{00,11\}, \quad R(0,2) = \{0000,1111\}.$$

$$R(1,1) = \{00, 01, 10, 11\} = \mathbb{F}_2^2.$$

Como o código de Reed-Muller é linear, podemos defini-lo ainda através de uma matriz geradora. Pela descrição acima, a matriz geradora para $R(0,m)$ é $G(0,m) = [1\ 1\ \cdots 1\ ]$ e a matriz geradora para $R(m,m)$ é $G(m,m) = I_{2^m}$. De forma geral,

Definição 1.19 *Uma **matriz geradora** $G(r,m)$ para o código de Reed Muller $R(r,m)$ é dada por*

$$G = \begin{bmatrix} G(0,m) \\ G(1,m) \\ \vdots \\ G(r,m) \end{bmatrix}.$$

Em particular: $G(0,m)$ é um vetor de comprimento $n = 2^m$ contendo somente $1's$, $G_1 = G(1,m)$ é uma matriz $m \times 2^m$ tendo como colunas todas as $m-$uplas binárias e $G(l,m)$ é construída de $G(1,m)$ considerando todos os produtos das linhas de $G(1,m)$. Por simplicidade de representação da matriz $G(1,m)$, considera-se a coluna mais à esquerda como sendo toda nula, a coluna mais à direita com todas as entradas iguais a 1 e as intermediárias como sendo todas $m-$uplas binárias, representando um inteiro, em ordem crescente.

O produto das linhas de uma matriz $G(r,m)$ é definido da seguinte forma: sejam a e b dois vetores tais que $a = (a_0, a_1, \ldots, a_{n-1})$ e $b = (b_0, b_1, \ldots, b_{n-1})$, então o produto entre a e b é definido componente a componente, isto é, $a \cdot b = (a_0b_0, a_1b_1, \ldots, a_{n-1}b_{n-1})$.

Definição 1.20 *Os **parâmetros** do $R(r,m)-$código de Reed Muller são*

$$\left[2^m, \binom{m}{0} + \binom{m}{1} + \ldots + \binom{m}{r}, 2^{m-r}\right].$$

Exemplo 1.2 *Considere $m = 4, n = 2^4 = 16$ e $r = 3$. Então,*

$$G(0,4) = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ ] = [a_0]$$

$$G(1,4) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}.$$

Como $G(1,4)$ tem 4 linhas, então $G(2,4)$ terá $\binom{4}{2} = 6$ linhas, isto é,

$$G(2,4) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_1a_2 \\ a_1a_3 \\ a_1a_4 \\ a_2a_3 \\ a_2a_4 \\ a_3a_4 \end{bmatrix}.$$

A matriz $G(3,4)$ *tem* $\binom{4}{3} = 4$ *linhas,*

$$G(3,4) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_1a_2a_3 \\ a_1a_2a_4 \\ a_1a_3a_4 \\ a_2a_3a_4 \end{bmatrix}.$$

Logo, a matriz geradora do código de Reed-Muller de terceira ordem e de comprimento 16 *é uma matriz* 15×16 *dada por*

$$G = \begin{bmatrix} G(0,4) \\ G(1,4) \\ G(2,4) \\ G(3,4) \end{bmatrix}.$$

1.1.5 Códigos de Reed-Muller de Primeira Ordem

Descrevemos aqui os códigos de Reed-Muller de primeira ordem, $R(1,m)$ definidos sobre $\mathbb{F}_2$, com base na sua matriz geradora. Considere todos os elementos de $\mathbb{F}_2^m$. Arrumando-os como vetores colunas em uma matriz A_m, $m \times 2^m$, de forma que o bloco $m \times 2^{m-1}$ à esquerda seja a matriz H_m do Código de Hamming e o vetor nulo de $\mathbb{F}_2^m$ na última coluna, ou seja,

$$A_m = (H_m, 0).$$

Construímos uma matriz G com $(m+1)$ linhas e 2^m colunas, em que sua primeira linha tem todas as entradas iguais a 1 e, abaixo, a matriz A_m em bloco, ou seja,

$$G = \begin{pmatrix} 1 & 1 \\ H_m & 0 \end{pmatrix}.$$

A matriz G é a matriz geradora do $R(1,m)-$código de Reed-Muller.

Proposição 1.7 *[18, Proposição 7.5] Os parâmetros do código* $R(1,m)$ *são* $[2^m; m+1; 2^{m-1}]$.

Demonstração: Como G tem 2^m colunas, é claro que o comprimento do código é 2^m. As $(m+1)$ linhas de G são linearmente independentes, devido ao bloco H_m e à primeira linha, logo a dimensão do código $R(1,m)$ é $(m+1)$.

Para mostrarmos que a distância mínima de $R(1,m)$ é 2^{m-1}, considere todas as palavras do código $R(1,m)$, exceto $u = 1 \ldots 1$, que tem peso 2^m.

Seja $c = v_{i_1} + \cdots + v_{i_r}$ uma palavra qualquer de $R(1,m)$, com os v_{i_j} os vetores linhas de G. Suponha, primeiro, que nenhum desses vetores é o vetor $1 \cdots 1$ e considere a matriz

$$B = \begin{pmatrix} v_{i_1} \\ \vdots \\ v_{i_r} \end{pmatrix}.$$

A matriz B possui 2^r colunas distintas, cada uma repetida 2^{m-r} vezes. Portanto, metade das colunas de A_m são de peso par e metade de peso ímpar.

Com isso, o vetor c possui metade de suas componentes iguais a zero e a outra metade iguais a 1. Logo o peso de c é 2^{m-1}.

Se um dos $v_i's$ for o vetor $1 \dots 1$, como a soma dos outros $v_i's$ tem metade de suas entradas iguais a 1, segue o mesmo resultado.

■

1.2 Códigos encurtados

Os códigos de bloco podem ser modificados alterando a sua dimensão e comprimento. Isto pode ser feito por diversas razões, como por exemplo, quando os valores de n e k não são mais convenientes (não servem para preencher uma trama de dados de tamanho normalizado, por exemplo) ou simplesmente para melhorar o desempenho do código. Para auxiliar nos exemplos, descrevemos uma modificação em códigos denominada **códigos encurtados**. Para esta seção veja [1].

Os códigos encurtados são obtidos dos originais removendo um ou mais bits de informação. O comprimento do código fica menor mas o número de bits de paridade se mantém.

Considerando o código sobre $\mathbb{F}_2$ e as 2^k palavras do código original, o que se faz é forçar a zero alguns (l) bits de informação e depois removê-los (não se transmitem). As palavras alteradas, aquelas cujos bits mudaram, desaparecem e fica-se com a estrutura

$$(n, k) \longrightarrow (n - l, k - l).$$

O novo código passa a ter 2^{k-l} palavras ao invés das 2^k originais (se $l = 1$ então $2^{k-1} = \frac{2^k}{2}$, ou seja, o código passa a ter metade das palavras) e a distância miníma é a mesma ou superior, dependendo dos bits removidos. Observe que a distância mínima não poderia ser menor porque as palavras que restam são um subconjunto das que já existiam removendo bits nulos e por isso têm o mesmo peso.

A remoção dos bits de ordem i (com $i = 1, 2, \cdots, k$) é equivalente a eliminar as linhas e as colunas de ordem i da matriz geradora e a eliminar as colunas de ordem i da matriz teste de paridade; esta permanece com o mesmo número de linhas porque o número de bits de paridade não foi alterado. Por este motivo, o número de erros corrigíveis nos códigos encurtados é o mesmo dos códigos donde estes derivam.

Não existe uma regra geral que indique quais são os *bits* de informação que devem ser removidos. O que se faz por simplicidade e conveniência é remover um ou mais bits de informação consecutivos.

Exemplo 1.3 *Considere o* $[7, 4]$ *código de Hamming com matriz geradora* G *e matriz teste de paridade* H *dadas por*

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Vamos determinar o código de Hamming encurtado $[6, 3]$ *removendo o terceiro bit de informação.*

A nova matriz geradora G' obtém-se da matriz G eliminando a 3º linha e a 3º coluna:

$$G' = \left[\begin{array}{cc|c|cccc} 1 & 0 & \mathbf{0} & 0 & 1 & 1 & 0 \\ 0 & 1 & \mathbf{0} & 0 & 1 & 0 & 1 \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \hline 0 & 0 & \mathbf{0} & 1 & 1 & 1 & 1 \end{array}\right] = \left[\begin{array}{cccccc} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right],$$

e a nova matriz teste de paridade H', é obtida da matriz teste de paridade H removendo-lhe a terceira coluna. Assim,

$$H' = \left[\begin{array}{cc|c|cccc} 1 & 1 & \mathbf{0} & 1 & 1 & 0 & 0 \\ 1 & 0 & \mathbf{1} & 1 & 0 & 1 & 0 \\ 0 & 1 & \mathbf{1} & 1 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{cccccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right].$$

Capítulo 2

Tópicos sobre redes neurais e grafos

Neste capítulo descrevemos alguns conceitos que serão necessários ao longo do trabalho sobre redes neurais e grafos. Para a Seção 2.1, exceto quando especificado, as principais referências são [17], [32], [41] e para a Seção 2.2 sobre grafos [5].

2.1 Redes neurais

A estrutura das redes neurais artificiais foi desenvolvida a partir de modelos conhecidos de sistemas nervosos biológicos e do próprio cérebro humano [41, p. 33]. De acordo com [17, p. 28] uma rede neural se assemelha ao cérebro em dois aspectos:

1. o conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem;
2. as forças de conexão entre os neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.

Para estabelecer esta relação entre as redes neurais artificiais e o cérebro humano, descrevemos nesta seção alguns aspectos do neurônio biológico, que é a célula elementar do sistema nervoso cerebral, e um modelo para o neurônio artificial. Uma síntese sobre as funções de ativação, a arquitetura, o aprendizado e um breve histórico das redes neurais também serão feitos.

2.1.1 O cérebro humano e o neurônio biológico

Com o intuito de estabelecer a relação entre o neurônio biológico e um neurônio artificial, descrevemos aqui o modo pelo qual os sistemas biológicos executam o processamento de informações.

Nos seres vivos, as células, menores unidades estruturais e funcionais, agrupam-se formando tecidos, que por sua vez, agrupam-se em órgãos. Dentre os quatro tipos básicos de tecidos, segundo as características morfológicas[1] e as propriedades funcionais, está o **tecido nervoso** [32].

O tecido nervoso encontra-se distribuído pelo organismo, porém interligado, resultando no **sistema nervoso**. Ele forma órgãos como o encéfalo e a medula espinhal,

[1]Em Biologia, morfologia é o estudo da configuração e da estrutura externa de um órgão ou ser vivo.

que compõem o **sistema nervoso central**, Figura 2.1. O tecido nervoso localizado além do sistema nervoso central é denominado **sistema nervoso periférico** e é constituído por aglomerados de neurônios, os **gânglios nervosos**, e por feixes de prolongamentos dos neurônios, os **nervos** [23].

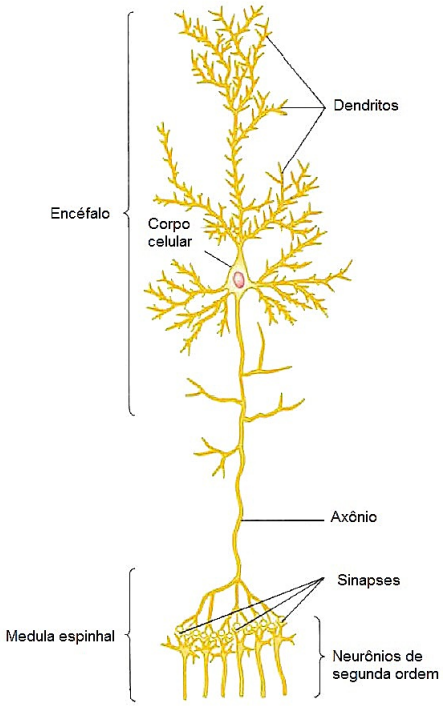


Figura 2.1: Estrutura de um neurônio grande, presente no encéfalo, no qual estão apontadas suas partes funcionais mais importantes (Redesenhado a partir de [15]).

A célula elementar do sistema nervoso cerebral é o **neurônio** [41, p. 29]. Tal elemento biológico possui um corpo celular com o núcleo e outras organelas e do qual partem os prolongamentos, que são os **dendritos** e o **axônio**.

Os neurônios formam uma rede de conexões capaz de captar informações dos receptores sensoriais, processar estas informações, originar uma memória e gerar os sinais apropriados para as células efetoras[2] [27, p. 84]. Assim, seu papel se resume a conduzir impulsos (estímulos elétricos provenientes de reações físico-químicas) sob determinadas condições de operação. Estes impulsos, usualmente conhecidos como **potenciais de ação** ou **pulsos**, originam-se no corpo celular dos neurônios, ou perto dele e, então, se propagam através dos neurônios individuais a velocidade e amplitude constantes.

Portanto, a informação é transmitida no sistema nervoso central principalmente na forma de potenciais de ação, que podem ser chamados simplesmente de "impulsos nervosos" que se propagam por uma sucessão de neurônios, um após o outro [15, p. 559].

Descrevendo as três partes principais dos neurônios, Figura 2.2, temos:

1. **Dendritos**[3]: constituídos por vários finos prolongamentos que formam a árvore dendrital, sua principal função consiste em captar, de forma contínua, os estímulos do meio ambiente, de células epiteliais sensoriais ou de outros neurônios [23].

[2]São denominados efetores os tecidos, órgãos ou células que exercem uma ação ou uma atividade como resposta a um estímulo.

[3]Do grego *dendrites*, referente a árvores.

2. **Axônio**[4]: constituído por um único prolongamento, eferente do neurônio, tem como função conduzir os impulsos elétricos para outros neurônios conectores ou para aqueles que se conectam diretamente com o tecido muscular (neurônios efetuadores). A sua terminação é também constituída de ramificações denominadas **terminações sinápticas**.

3. **Corpo celular**: responsável por processar todas as informações advindas dos dendritos a fim de produzir um **potencial de ativação** que indicará se o neurônio poderá disparar um impulso elétrico ao longo de seu axônio. No corpo celular encontram-se também as principais organelas citoplasmáticas (núcleo, mitocôndria, centríolo, lisossomo, etc.) do neurônio.

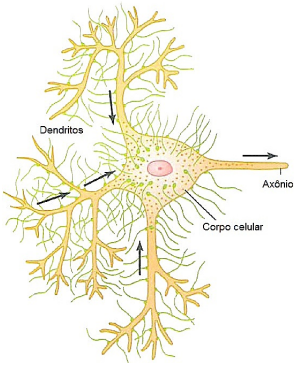


Figura 2.2: Neurônio, exibindo terminações pré-sinápticas no corpo celular e nos dendritos (Redesenhado a partir de [15]).

Os locais de contato entre dois neurônios ou entre um neurônio e a célula efetora, como uma célula glandular ou uma célula muscular, são as **sinapses** (do grego *synapsis*, conexão).

As sinapses são unidades estruturais e funcionais elementares que medeiam as interações entre os neurônios [17, p.32]. Desta forma, elas se configuram como as conexões que viabilizam a transferência de impulsos elétricos do axônio de um neurônio para os dendritos de outros. Porém, não há contato físico entre os neurônios na **junção (fenda) sináptica**, sendo que os elementos neurotransmissores liberados são os responsáveis por ponderar a transmissão de impulsos elétricos de um neurônio para o outro. A funcionalidade de um neurônio é dependente do comportamento dessas **ponderações sinápticas** que são dinâmicas e dependentes da química cerebral [19], Figura 2.3.

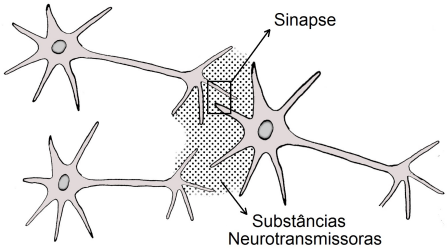


Figura 2.3: Conexões sinápticas entre neurônios (Redesenhado a partir de [41]).

Fazendo uma comparação, o sistema nervoso pode ser visto como um sistema de três estágios [17], como na Figura 2.4.

[4]Do grego *axon*, eixo.

Figura 2.4: Representação em diagrama em blocos do sistema nervoso [17].

O centro do sistema é o **cérebro**, representado pela rede neural, que recebe informações e executa a tomada de decisões. Os receptores convertem estímulos do corpo humano ou do ambiente externo em impulsos elétricos que transmitem informação para a rede neural (cérebro). Os atuadores convertem impulsos elétricos gerados pela rede neural em respostas discerníveis como saídas do sistema.

2.1.2 Neurônio artificial

Os elementos computacionais ou unidades processadoras, denominadas **neurônios artificiais**, são modelos bem simplificados dos neurônios biológicos. Assim, uma vez definidos alguns aspectos da anatomia e fisiologia dos neurônios biológicos, na Seção 2.1.1, faremos aqui a descrição de um modelo de neurônio artificial. Exceto quando explicitado, todas as informações contidas nesta seção são provenientes de [41, pp. 33, 34]

O primeiro modelo matemático para uma rede neural [30] foi proposto por McCulloch, um psiquiatra e neuroanatomista por treinamento, e pelo matemático Pitts em 1943 [17]. Tal modelo é ainda o mais utilizado nas diferentes arquiteturas de redes neurais artificiais.

Nesta representação, que será detalhada abaixo, cada neurônio pode ser implementado conforme mostra a Figura 2.5.

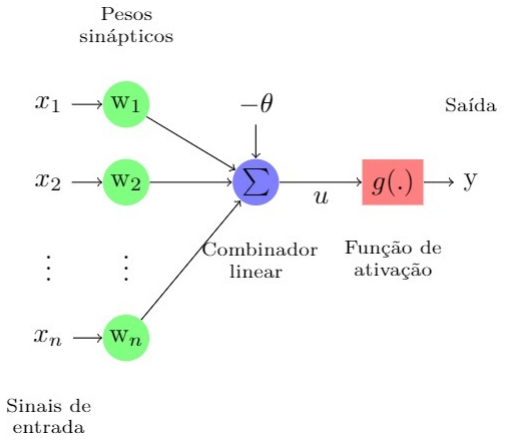


Figura 2.5: Neurônio artificial.

Observe que tal modelo de neurônio artificial pode ser comparado ao neurônio biológico como na Figura 2.6.

Assim como no neurônio biológico, no neurônio artificial as informações podem ser recebidas através de sensores ou outros neurônios artificiais que fazem parte da rede neural artificial. Estes sinais são processados e enviados para a saída.Desta forma, considerando a Figura 2.5, o neurônio artificial é constituído de sete elementos básicos:

1. **Sinais de entrada** $\{x_1, x_2, \ldots, x_n\}$: análogos aos impulsos elétricos externos

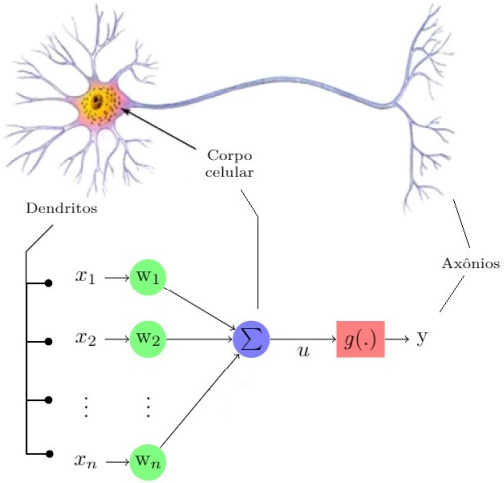


Figura 2.6: Comparação de um modelo de neurônio artificial com um neurônio biológico (Redesenhado a partir de [9]).

captados pelos dendritos no neurônio biológico, os sinais de entrada são os sinais ou medidas provenientes do meio externo e que representam os valores assumidos pelas variáveis de uma aplicação específica.

2. **Pesos sinápticos** $\{w_1, w_1, \ldots, w_n\}$**:** os pesos sinápticos, ou simplesmente pesos, são os valores que ponderam cada uma das variáveis de entrada da rede, permitindo quantificar as suas relevâncias em relação à funcionalidade do respectivo neurônio, ou seja, eles representam o grau de importância que determinada entrada possui em relação a um determinado neurônio.

   Em comparação com o modelo biológico, os pesos sinápticos no neurônio artificial representam as ponderações exercidas pelas junções sinápticas do modelo biológico e, ao contrário de uma sinapse do cérebro, o peso sináptico de um neurônio artificial pode estar em um intervalo que inclui valores negativos e positivos [17].

   Matematicamente, os pesos são vistos como um vetor $(w_1, w_2, \ldots, w_n)$ para um neurônio ou como uma matriz de pesos para um conjunto de neurônios.

3. **Combinador linear** $(\sum)$: sua função é somar sinais de entradas, que foram ponderados pelos respectivos pesos sinápticos a fim de produzir um valor de potencial de ativação.

4. **Limiar de ativação** (θ): é uma variável que especifica qual será o patamar apropriado para que o resultado produzido pelo combinador linear possa gerar um valor de disparo em direção à saída do neurônio.

5. **Potencial de ativação** (u): é o resultado obtido pela diferença do valor produzido entre o combinador linear e o limiar de ativação. Se tal valor é positivo, ou seja, se $u \geq 0$, então o neurônio produz um potencial excitatório; caso contrário, o potencial será inibitório, isto é, se o valor obtido atingiu o limiar ele é transmitido adiante através da saída, caso contrário o sinal não é transferido.

6. **Função de ativação** (g): seu objetivo é limitar a saída do neurônio dentro de um intervalo de valores razoáveis a serem assumidos pela sua própria imagem funcional.

A função de ativação é também chamada de *função restritiva* [17, p.37] uma vez que restringe (limita) o intervalo permissível de amplitude do sinal de saída de um neurônio a um valor finito. Tipicamente, o intervalo normalizado da amplitude da saída de um neurônio é escrito como o intervalo unitário fechado $[0, 1]$ ou alternativamente $[-1, 1]$.

7. **Sinal de saída** (y): consiste do valor final produzido pelo neurônio em relação a um determinado conjunto de sinais de entrada, podendo ser também utilizado por outros neurônios que estão sequencialmente interligados.

As duas expressões seguintes sintetizam o resultado produzido pelo neurônio artificial proposto por McCulloch e Pitts em [30]

$$u = \sum_{i=1}^{n} w_i \cdot x_i - \theta \tag{2.1}$$

$$y = g(u) \tag{2.2}$$

As funções de ativação podem ser divididas em dois grupos principais, como funções parcialmente diferenciáveis e funções totalmente diferenciáveis, considerando-se todo o domínio de definição das mesmas. Para compreensão dos exemplos que serão feitos nas seções seguintes será suficiente aqui descrever as principais funções de ativação parcialmente diferenciáveis e a função linear que é uma função totalmente diferenciável. Para mais informações veja [41, p. 36].

Funções de ativação parcialmente diferenciáveis

As funções de ativação parcialmente diferenciáveis são aquelas que possuem pontos cujas derivadas de primeira ordem são inexistentes. As três principais funções de ativação nesta classe são a função degrau, a função degrau bipolar e a função rampa simétrica.

O resultado produzido pela aplicação de cada uma destas funções no potencial de ativação u do neurônio assumirá valores positivos, negativos ou nulos de acordo com os valores assumidos por u.

a) **Função degrau:** definida por

$$g(u) = \begin{cases} 1, & \text{se } u \geqslant 0, \\ 0, & \text{se } u < 0. \end{cases} \tag{2.3}$$

b) **Função degrau bipolar ou função sinal:** definida por

$$g(u) = \begin{cases} 1, & \text{se } u > 0, \\ 0, & \text{se } u = 0, \\ -1, & \text{se } u < 0. \end{cases} \tag{2.4}$$

Ainda, em problemas envolvendo classificação de padrões, a função degrau bipolar pode ser também aproximada pela expressão:

$$g(u) = \begin{cases} 1, & \text{se } u \geqslant 0, \\ -1, & \text{se } u < 0. \end{cases} \tag{2.5}$$

c) **Função rampa simétrica:** o resultado produzido pela aplicação desta função são iguais aos próprios valores dos potenciais de ativação quando estes estão definidos no intervalo $[-a, a]$, restringindo-se aos valores limites caso contrário:

$$g(u) = \begin{cases} a, & \text{se } u > a, \\ u, & \text{se } -a \leq u \leq a, \\ -a, & \text{se } u < a. \end{cases} \tag{2.6}$$

Função linear: função de ativação totalmente diferenciável

Também chamada de função identidade, a função linear produz resultados de saída idênticos aos valores do potencial de ativação (u), desta forma:

$$g(u) = u. \tag{2.7}$$

2.1.3 Arquiteturas de redes neurais artificiais

A arquitetura de uma rede neural artificial define a forma como os neurônios da rede podem ser agrupados. Tal organização é estruturada através do direcionamento das conexões sinápticas dos neurônios. Uma rede neural artificial pode ser dividida em três partes, denominadas camadas [41, p. 45]. São elas:

1. **Camada de entrada:** é a camada responsável pelo recebimento de informações, dos dados de entrada que geralmente são normalizados para se obter uma melhor precisão numérica frente às operações matemáticas realizadas pela rede.

2. **Camadas escondidas, intermediárias, ocultas ou invisíveis:** são as camadas compostas por neurônios que extraem as características associadas ao processo ou sistema a ser inferido. Quase todo o processamento interno da rede é realizado nessas camadas.

3. **Camada de saída:** também constituída de neurônios, é a camada responsável pela produção e apresentação dos resultados finais da rede, os quais são provenientes dos processamentos efetuados pelos neurônios das camadas anteriores.

Quanto ao tipo de redes, as arquiteturas de redes neurais artificiais podem ser divididas em três classes fundamentalmente diferentes: **redes *feedforward* de camada simples**, Figura 2.7, **redes *feedforward* de camadas múltiplas**, Figura 2.8, e **redes recorrentes**, Figura 2.9. A diferença entre elas está no fato de que nas redes do tipo *feedforward* o fluxo de informação segue da entrada para a saída enquanto que nas redes realimentadas ou recorrentes as saídas se comunicam com a camada de entrada.

Ainda, na arquitetura *feedforward* de camada simples tem-se apenas uma camada de entrada e uma única camada de neurônios que consiste da camada de saída. Já as redes *feedforward* de camadas múltiplas são constituídas pela presença de uma ou mais camadas escondidas de neurônios.

Entre os principais tipos de redes que possuem realimentação está a rede de Hopfield e entre os principais tipos de redes com arquitetura *feedforward* de camada simples e camadas múltiplas está a rede *Perceptron*. Falaremos a respeito de tais redes nas seções seguintes com o objetivo de tornar mais claro alguns dos exemplos.

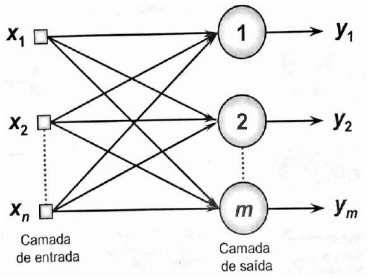


Figura 2.7: Exemplo de rede *feedforward* de camada simples [41].

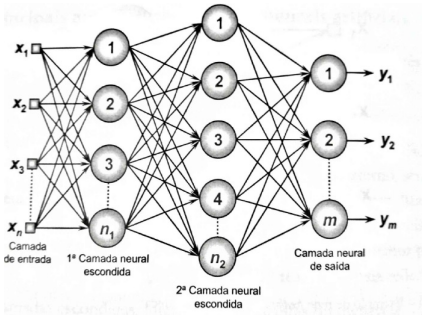


Figura 2.8: Exemplo de rede *feedforward* de camadas múltiplas [41].

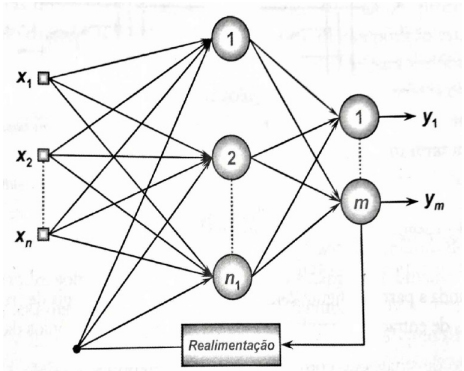


Figura 2.9: Exemplo de rede recorrente [41].

2.1.4 Aprendizagem das redes neurais

Dentre as principais características das redes neurais está a capacidade de **aprendizagem da rede**. A **aprendizagem** é um processo pelo qual os parâmetros livres de uma rede neural são adaptados através de um processo de estimulação pelo ambiente no qual a rede está inserida [17, p.75].

A rede neural é capaz de aprender por meio de exemplos a partir da apresentação de amostras (padrões) que exprimem o comportamento do sistema. Uma vez que a rede aprendeu como as entradas e saídas se relacionam, ela é capaz de generalizar soluções e assim produzir uma saída próxima da esperada independente dos sinais inseridos em suas entradas.

O processo de treinamento de uma rede neural consiste da aplicação dos **algoritmos de aprendizagem** (passos ordenados que ajustam seus pesos sinápticos e limiares de ativação) e o tipo de aprendizagem é determinado pela maneira pela qual a modificação dos parâmetros da rede ocorre [17].

Dentre os processos de treinamento das redes neurais, tem-se o **treinamento supervisionado**. Neste processo, para cada amostra dos sinais de entrada, tem-se disponível as respectivas saídas desejadas. Assim, os pesos sinápticos e limiares são continuamente ajustados mediante a aplicação de ações comparativas executadas pelo algoritmo de aprendizagem que supervisionam a defasagem entre as respostas produzidas pela rede em relação às respostas desejadas. Esta diferença é utilizada no processo de ajuste.

A rede será considerada **treinada** quando tal defasagem estiver dentro de valores

aceitáveis. Assim, os parâmetros livres da rede são ajustados em função de se conhecer primeiramente quais são as saídas desejadas ao sistema investigado [41, p. 52].

2.1.5 Notas históricas

Nesta seção, com o intuito de ampliar a compreensão a cerca de redes neurais e introduzir o modelo de redes *Perceptron* e o modelo de Hopfield, que serão discutidos nos próximos capítulos, faremos um resumo histórico sobre as redes neurais artificiais, descrevendo algumas das principais contribuições para o desenvolvimento desta teoria. Para esta seção, exceto quando explicitado, as principais referências são [17] e [41].

A primeira publicação relacionada às redes neurais é devida ao psiquiatra e neuroanatomista McCulloch e ao matemático Pitts. No artigo publicado em 1943, *A Logical Calculus of the Ideas Immanent in Nervous Activity* [30], McCulloch e Pitts descreveram o primeiro modelo matemático inspirado no neurônio biológico resultando na concepção do neurônio artificial.

O modelo descrito era um neurônio simples, um dispositivo binário do qual a saída poderia ser *pulso* ou *não pulso*. Com este dispositivo eles implementaram funções booleanas [24] em que a atividade do único neurônio de saída era o valor verdade da operação lógica binária representada nos neurônios de entrada [2].

A essência da proposta de McCulloch e Pitts foi a seguinte: "a inteligência é equivalente ao cálculo de predicados que por sua vez pode ser implementado por funções booleanas. Por outro lado, o sistema nervoso é composto de redes de neurônios que, com as devidas simplificações, tem a capacidade básica de implementar estas funções booleanas. Conclusão: a ligação entre inteligência e atividade nervosa fica estabelecida de forma científica" [24, p. 29].

Em 1948 o matemático Wiener escreveu o livro *Cybernetics* que descrevia alguns conceitos importantes sobre controle, comunicação e processamento estatístico de sinais.

Em 1949, a publicação do livro *The Organization of Behavior*, do biólogo e psicólogo Donald Hebb, trouxe um desenvolvimento significativo para as redes neurais: o primeiro método de treinamento para redes neurais foi proposto e denominado **regra de aprendizado de Hebb**. Ele propôs que a conectividade do cérebro é continuamente modificada conforme um organismo vai aprendendo tarefas funcionais diferentes e que agrupamentos neurais são criados por modificações. Assim, seu postulado determina como ocorre a modificação das ligações sinápticas entre neurônios.

A partir daí, vários outros pesquisadores deram sequência ao estudo das redes neurais e ao trabalho de desenvolver modelos matemáticos fundamentados no neurônio biológico. Em ordem cronológica, podemos destacar:

1952 - Publicação do livro *Design for a Brain: The Origin of Adaptive Behavior* do médico neurologista, que também estudou matemática avançada, Ashby. Neste livro ele enfatizava os aspectos dinâmicos do organismo vivo como uma máquina e o conceito correlacionado de estabilidade.

1956 – Iniciou-se o trabalho sobre memória associativa por Taylor, que foi seguido por outros trabalhos relevantes de pesquisadores em 1961 e 1972. Em 1956 houve também a Primeira Conferência Internacional de Inteligência Artificial em que foi apresentado um modelo de rede neural artificial pelo pesquisador Nathaniel Rochester, que consistia em uma simulação de centenas de neurônios interconectados através de um sistema que

verifica como a rede responderia a estímulos ambientais.

1958 –Frank Rosenblatt, entre 1957 e 1958, desenvolveu o primeiro neurocomputador denominado *Mark I - Perceptron*. Rosenblatt propôs o modelo *Perceptron* de rede neural que é capaz de classificar padrões.

1960 - Widrow e Hoff desenvolveram um tipo de rede denominada *Adaline* (*Adaptive linear element*), elemento linear adaptativo, que se diferenciava do *Perceptron* pelo procedimentro de aprendizagem.

Todos estes trabalhos incentivaram muitos outros pesquisadores a realizar pesquisas nesta área até a publicação de 1969:

1969 – No livro *Perceptrons – an introduction to computational geometry*, Minsky e Papert demonstraram que existiam limites fundamentais para o que os perceptrons de uma única camada poderiam calcular. Foi demonstrado que tais redes tinham limitações em aprender o relacionamento entre as entradas e saídas de funções lógicas bem simples como o X_{OR} (ou-exclusivo), mais especificamente, houve a demonstração da impossibilidade das redes neurais realizarem a correta classificação de padrões para classes não linearmente separáveis.

Devido a esta publicação, houve uma significativa queda nas pesquisas e nos investimentos relacionados às redes neurais. Apenas alguns poucos pesquisadores continuaram e, embora fossem obtidas algumas contribuições que datam após 1969, o interesse pelas redes neurais só foi retomado de maneira significante quando o físico e biólogo John Hopfield propôs um modelo com propriedades de memória associativa, semelhante ao anteriormente desenvolvido por Little [25] (1974).

1982- John Hopfield trouxe o rigor matemático da Mecânica Estatística ao campo das redes neurais artificiais, fazendo analogias destas a sistemas físicos [10]. Além disto, Hopfield utilizou a ideia de uma função de energia para formular um novo modo de se entender a computação executada por redes recorrentes com conexões sinápticas simétricas. Estas redes foram denominadas *redes de Hopfield*. Embora não sejam modelos realísticos dos sistemas neurobiológicos, as redes de Hopfield estão vinculadas ao princípio de armazenamento de informação em redes dinamicamente estáveis.

O artigo de Hopfield de 1982, *Neural networks and physical systems with emergent collective computational abilities* [20], é considerado uma das publicações mais influentes responsáveis pelo ressurgimento do interesse na área de redes neurais. A outra publicação considerada tão importante quanto é atribuída à Rumelhart e McClelland.

1986- Rumelhart, Hilton e Williams, redescobrem o algoritmo de aprendizado *Backpropagation*, inicialmente introduzido por Werbos [42] (1974). Eles mostram como treinar uma rede estática de múltiplas camadas, solucionando inclusive o antigo problema de aprendizado dos padrões da função lógica X_{OR}. Neste mesmo ano é então publicado o livro *Parallel distributed processing* editado por Rumelhart e McClelland [39]. A proposição do algoritmo *Backpropagation* impulsionou inúmeras pesquisas na área nas décadas seguintes.

2.2 Teoria dos grafos

Nesta seção definimos alguns conceitos e apresentamos alguns resultados da Teoria de Grafos que serão necessários nos capítulos subsequentes. As principais referências são [3], [5] e [35].

Definição 2.1 *Um* ***grafo*** $\hat{G}$ *é um par ordenado* $(\hat{V}, \hat{E})$ *consistindo de um conjunto* $\hat{V}$ *de* ***vértices*** *e um conjunto* $\hat{E}$ *de* ***arestas****, disjunto de* $\hat{V}$*, junto com uma função de incidência* $\psi_{\hat{G}}$ *que associa a cada aresta de* $\hat{G}$ *um par não ordenado de vértices (não necessariamente distintos) de* $\hat{G}$*.*

Definição 2.2 *Sejam "e" uma aresta e* u *e* v *vértices tais que* $\psi_{\hat{G}}(e) = \{u, v\}$*. Dizemos que "e"* ***une*** u *e* v *e os vértices* u *e* v *são chamados de* ***extremidades*** *de* e*.*

Definição 2.3 *Um* ***grafo orientado*** $\hat{G}$ *é um par ordenado* $(\hat{V}, \hat{A})$ *consistindo de um conjunto* $\hat{V}$ *de vértices e um conjunto* $\hat{A}$*, disjunto de* $\hat{V}$*, de* ***arcos*** *juntamente com uma função de incidência* $\psi_{\hat{G}}$ *que associa a cada arco de* $\hat{G}$ *um par ordenado de vértices, não necessariamente distinto, de* $\hat{G}$*.*

Definição 2.4 *Sejam* a *um arco e* $\psi_{\hat{G}}(a) = (u, v)$*. Dizemos que* a *une* u *e* v*, com* u *o vértice inicial e* v *o vértice final, sendo estes as duas extremidades de* a*.*

Quando a orientação de um arco é irrelevante para a discussão, nos referiremos ao arco como uma aresta do grafo orientado. Por conveniência, algumas vezes, nos referiremos também aos vértices do grafo como **nós**.

Assim, intuitivamente, um grafo orientado, que é também chamado de *dígrafo*, é formado por vértices conectados por arestas direcionadas ou arcos.

Graficamente, um grafo orientado é representado como na Figura 2.10 em que cada arco é representado por uma flecha de u para v indicando a orientação. Por outro lado, um **grafo não orientado** é um grafo em que todas as arestas são bidirecionais, Figura 2.11.

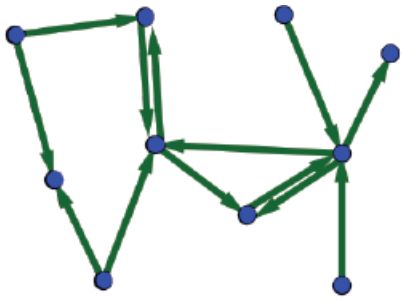


Figura 2.10: Grafo orientado.

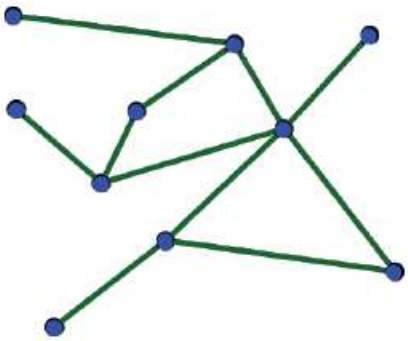


Figura 2.11: Grafo não orientado.

Definição 2.5 *Um* ***grafo ponderado*** *é um grafo* $\hat{G}$ *juntamente com uma função que associa um número real* $W(e)$ *(usualmente não negativo) a cada aresta* e*. Esse número é chamado* ***peso****.*

O peso associado a uma aresta (i, j) é representado pela notação W_{ij}.

Exemplo 2.1 *Considere o conjunto de vértices* $\hat{V} = \{1, 2, 3, 4\}$ *e o conjunto de arestas* $\hat{E} = \{(1,2), (1,4), (4,3), (2,3), (2,4)\}$ *cujos pesos associados são* $W_{12} = 2, W_{14} = 5, W_{23} = 1, W_{24} = 1$ *e* $W_{43} = 3$*. O grafo apresentado na Figura 2.12 é um grafo orientado e ponderado.*

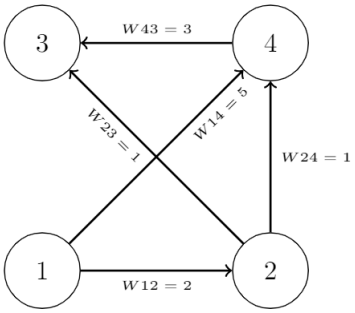


Figura 2.12

Definição 2.6 *Um subconjunto $\hat{V}_1$ do conjunto de vértices $\hat{V}$ de um grafo $\hat{G}$ é chamado de* ***separador de vértices*** *de $\hat{G}$ se cada aresta de $\hat{G}$ ou une um par de vértices em $\hat{V}_1$ ou une um par de vértices em $\hat{V} - \hat{V}_1$.*

Da definição, o conjunto vazio e o próprio conjunto $\hat{V}$ são separadores de vértices triviais.

Definição 2.7 *Seja $\hat{V}_1$ um subconjunto próprio não vazio do conjunto de vértices $\hat{V}$ de um grafo $\hat{G}$. O conjunto de todas as arestas do grafo que unem vértices de $\hat{V}_1$ com vértices em $\hat{V} - \hat{V}_1$ é chamado de* ***corte*** *gerado pelo subconjunto de vértices $\hat{V}_1$.*

Observe que $\hat{V}_1$ e $\hat{V}_{-1} = \hat{V} - \hat{V}_1$ definem o mesmo corte e que qualquer coleção de vértices isolados define um ***corte trivial (vazio)****.*

Um subconjunto próprio não vazio $\hat{V}_1$ de vértices não isolados de um grafo pode também definir um corte vazio, mesmo se o seu complemento contiver vértices não isolados. Por exemplo, observe no grafo da Figura 2.13, o subconjuto $\hat{V}_1 = \{v_8, v_7\}$ de vértices do grafo define um corte vazio pois não existem arestas unindo $\hat{V}_1 = \{v_8, v_7\}$ a $\hat{V} - \hat{V}_1$. Pela

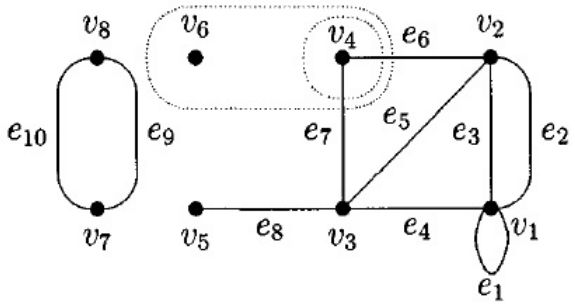


Figura 2.13: [35]

definição, podemos observar ainda que o corte definido por um subconjunto próprio não vazio de vértices $\hat{V}_1$ é um corte vazio se, e somente se, $\hat{V}_1$ é um separador de vértices. Assim, o próprio conjunto $\hat{V}$ de vértices do grafo, que é um separador de vértices trivial, define um corte trivial (vazio).

Exemplo 2.2 *Considere o grafo de 5 nós e 8 arestas da Figura 2.14. Se $\hat{V}_1 = \{2\}$, então $\hat{V}_{-1} = \hat{V} - \hat{V}_1$. Logo, $\hat{V}_{-1} = \{1, 3, 4, 5\}$.*

Assim, o conjunto das arestas $\{(1, 2), (2, 5), (2, 4)\}$ é um corte no grafo $\hat{G}$, Figura 2.14.

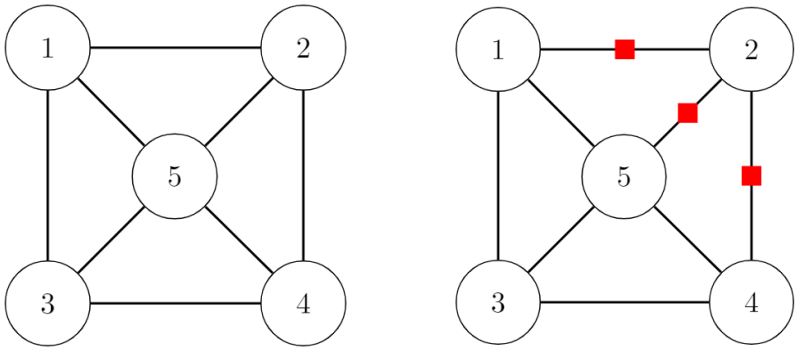


Figura 2.14: Exemplo de cortes em um grafo.

Definição 2.8 *O **peso de um corte** é a soma do peso de suas arestas.*

Definição 2.9 *Um **corte mínimo (MC)** de um grafo é um corte com peso mínimo.*

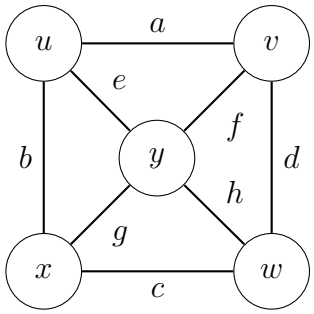
Definição 2.10 *Duas **arestas** são **adjacentes** se possuem um vértice em comum e, uma **aresta** é **incidente ao vértice** quando este for uma de suas extremidades.*

Definição 2.11 *Seja $\hat{G} = (\hat{V}, \hat{E})$ um grafo com n vértices e m arestas.*

*i) a **matriz de incidência** de $\hat{G}$ é a matriz $M_{\hat{G}} := (m_{ve})$, $n \times m$, em que m_{ve} é o número de vezes que o vértice v e a aresta e são incidentes.*

*ii) a **matriz de adjacência** de um grafo $\hat{G}$ é uma matriz $A_{\hat{G}} := (a_{uv})$, $n \times n$, em que a_{uv} é o número de arestas unindo os vértices u e v.*

Exemplo 2.3 *Considere o seguinte grafo de 5 nós e 8 arestas:*



A matriz de incidência deste grafo é dada por $M_{\hat{G}} := (m_{ve})$ e a matriz de adjacência é dada por $A_{\hat{G}} := (a_{uv})$, de acordo com as tabelas a seguir.

	a	b	c	d	e	f	g	h
u	1	1	0	0	1	0	0	0
v	1	0	0	1	0	1	0	0
w	0	0	1	1	0	0	0	1
x	0	1	1	0	0	0	1	0
y	0	0	0	0	1	1	1	1

$$\Rightarrow M_{\hat{G}} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\begin{array}{c|ccccc} & u & v & w & x & y \\ \hline u & 0 & 1 & 0 & 1 & 1 \\ v & 1 & 0 & 1 & 0 & 1 \\ w & 0 & 1 & 0 & 1 & 1 \\ x & 1 & 0 & 1 & 0 & 1 \\ y & 1 & 1 & 1 & 1 & 0 \end{array} \Rightarrow A_{\hat{G}} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Observação 2.1 *Para um grafo orientado podemos definir a matriz de incidência como uma matriz $n \times m$ tal que cada entrada é dada por*

$$m_{ve} = \begin{cases} 1 \ \textit{se a aresta} \ e_j \ \textit{sai do vértice} \ v_i, \\ -1 \ \textit{se a aresta} \ e_j \ \textit{chega do vértice} \ v_i, \\ 0 \ \textit{caso contrário.} \end{cases}$$

Definição 2.12 *Um grafo $\hat{G}$ é um* ***grafo conexo*** *se, para cada partição de seu conjunto de vértices em dois conjuntos não-vazios X e Y, houver uma aresta com uma extremidade em X e uma extremidade em Y, ou seja, $\hat{G}$ é conexo se existir um caminho entre qualquer par de vértices de $\hat{G}$, como na Figura 2.15, caso contrário, o grafo $\hat{G}$ é desconexo, veja Figura 2.16.*

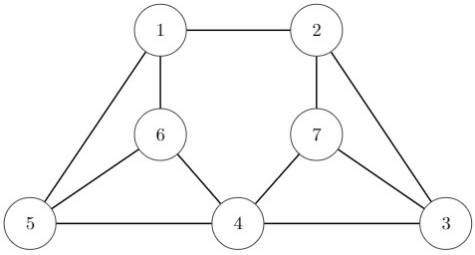


Figura 2.15: Grafo conexo.

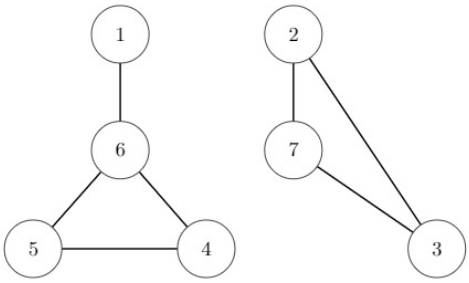


Figura 2.16: Grafo desconexo.

As seguintes definições e a Proposição 2.1 serão utilizados para a compreensão de um resultado no Capítulo 3.

Definição 2.13

1. *Um* ***laço*** *é uma aresta tal que se $v \in \hat{V}$ então a aresta $\{v, v\}$ pertence a $\hat{E}$.*

2. *Chamamos de* ***grau de um vértice****, e denotamos por $g(v)$, o número de arestas que contém o vértice v. Se existirem laços incidentes no vértice v, cada laço contribuirá em duas unidades para o grau de v.*

3. *O* ***grau máximo*** *de um grafo, denotado por $\Delta(\hat{G})$ é dado por*

$$\Delta(\hat{G}) = max\{g(v)|v \in \hat{V}\},$$

isto é, dentre todos os vértices do grafo, o grau máximo é dado por aquele que possuí o maior número de arestas.

4. *Analogamente, o* ***grau mínimo*** *de um grafo $\hat{G}$, denotado por $\delta(\hat{G})$, é dado por*

$$\delta(\hat{G}) = min\{g(v)|v \in \hat{V}\}.$$

Proposição 2.1 *[4, Lema 3.17] Seja $\hat{G}$ um grafo com $\hat{V} = \{a_1, a_2, \ldots, a_n\}$ um conjunto de vértices cujo grau de cada vértice é dado por $g(a_1), g(a_2), \ldots, g(a_n)$. O número m de arestas em $\hat{G}$ é dado por:*

$$m = \left(\frac{g(a_1) + g(a_2) + \ldots + g(a_n)}{2} \right)$$

Em particular, a soma dos graus de $\hat{G}$ é um número par.

Demonstração: *De fato, cada vértice a_i fornece $g(a_i)$ e, como cada aresta contém exatamente dois vértices, devemos então tomar a metade da soma dos graus.*

■

Capítulo 3

Redes neurais e códigos corretores de erros

Neste capítulo estabelecemos uma relação entre redes neurais e códigos corretores de erros. O objetivo é mostrar ao final uma forma de como as redes neurais podem ser utilizadas para a decodificação de máxima verossimilhança de códigos corretores de erros. As principais referências para este capítulo são [3], [6], [7] e [8].

3.1 Redes neurais vistas como grafos

Definição 3.1 *Uma* ***rede*** *é um grafo cujos vértices e/ou arestas possuem valores associados.*

As redes podem representar muitos tipos de sistemas do mundo real. Por exemplo, a internet pode ser descrita como uma rede em que os nós são computadores, ou outros dispositivos, e as arestas são conexões físicas (ou sem fios) entre os dispositivos [36].

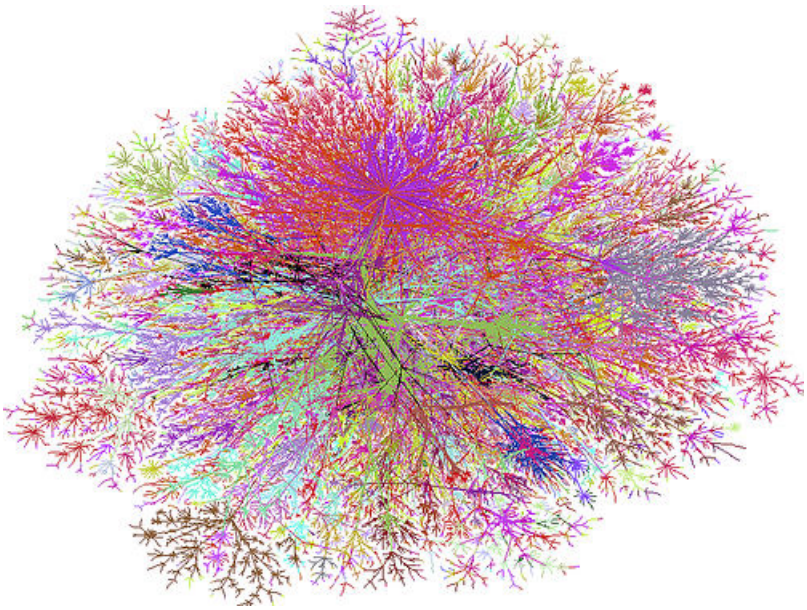


Figura 3.1: Rede de conexões entre dispositivos dentro da Internet [36]

A definição de uma rede neural, feita na Introdução, pode agora ser compreendida a luz dos conceitos de grafos feitos na Seção 2.2:

Definição 3.2 *As **redes neurais**, ou redes neurais artificiais, são modelos computacionais inspirados no sistema nervoso de seres vivos e possuem a capacidade de aquisição de conhecimento (baseado em informações).*

De forma análoga ao cérebro, uma rede neural é composta por várias unidades de processamento, correspondentes aos neurônios. Estas unidades são interconectadas através de pesos, que são valores numéricos, representando as sinapses[1]. As sinapses são responsáveis por determinar uma saída que servirá de entrada para outra unidade. Elas serão representadas aqui por vetores ou matrizes de pesos sinápticos.

Definição 3.3 *[33, pp. 41, 46.] Um **sistema** é um conjunto não vazio de objetos agrupados por alguma interação ou interdependência, de modo que existam relações de causa e efeito nos fenômenos que ocorrem com os elementos deste conjunto.*

*Um sistema é de **tempo discreto** se o tempo t, variável independente, é um número inteiro não negativo.*

*A **evolução** de um sistema de tempo discreto é governada por uma ou mais equações de diferenças (finitas), que é um tipo de equação que relaciona o valor de uma variável $x \in \mathbb{R}$ no instante t a valores de x em outros instantes como $t+1$, $t+3$, $t-2$, etc. A partir disto, podemos definir:*

Definição 3.4 *Um **modelo de rede neural** é um sistema de tempo discreto que pode ser representado por um grafo não orientado ponderado.*

Em um modelo de rede neural existe um peso associado a cada aresta do grafo e um valor limite associado a cada vértice (que pode ser chamado nó ou neurônio) do grafo.

Definição 3.5 *A **ordem** da rede neural é o número de nós no grafo correspondente.*

3.2 O Modelo de Hopfield

As redes de Hopfield possuem, dentre as suas características, a capacidade de memorização e armazenamento de informações e podem ser utilizadas em diversas aplicações como a reconstrução de imagens, classificação de objetos ou correção de erros.

Estas redes caracterizam um modelo de rede recorrente, cujos neurônios são todos interconectados de modo que as saídas estão ligadas às entradas por meio de um atraso de tempo. Deste modo, devido a recorrência, as redes de Hopfield possuem também características temporais implicando que a resposta da rede dependerá sempre do seu estado no intervalo de tempo anterior. Descreveremos nesta seção tal modelo de rede neural juntamente com sua propriedade de convergência, estabelecendo ainda a relação com o problema de encontrar o corte mínimo em um grafo.

Seja N uma rede neural de ordem n, então N é unicamente definida por (W, T) com

[1]Sinapses ou terminações nervosas são as estruturas pelas quais os neurônios estabelecem comunicações entre si, veja página 16.

1. W uma matriz $n \times n$ simétrica de diagonal zero e $W_{i,\,j}$ igual ao **peso associado à aresta** (i, j).
2. T um vetor de dimensão n, com T_i denotando o **limite associado ao nó** i.

Cada nó (neurônio) pode estar em um dos dois estados possíveis, 1 ou -1.

- O **estado do nó** i no momento t é denotado por $V_i(t)$.
- O **estado da rede neural** no tempo t é denotado pelo vetor $V(t)$.
- O **próximo estado de um nó** é calculado por

$$V_i(t+1) = sgn(H_i(t)) = \begin{cases} 1, & \text{se } H_i(t) \geqslant 0, \\ -1, & \text{caso contrário,} \end{cases} \tag{3.1}$$

  com

$$H_i(t) = \sum_{j=1}^{n} W_{ji} V_j(t) - T_i.$$

- O **próximo estado da rede**, isto é, $V(t+1)$ é calculado a partir do estado atual, realizando a avaliação (3.1) em um subconjunto dos nós da rede que será denotado por $S(t)$.
- Os **modos de operação** são determinados pelo metódo pelo qual o conjunto $S(t)$ é selecionado em cada intervalo de tempo.
- O modelo descrito em (3.1) é conhecido como **modelo de Hopfield.**

Definição 3.6 *1) Se o cálculo é realizado em um único nó em qualquer intervalo de tempo, isto é,* $|S(t)| = 1$*, então a rede está operando em* ***modo serial.***

2) Se $|S(t)| = n$*, então a rede está operando em um* ***modo totalmente paralelo.***

3) Todos os outros casos, $1 < |S(t)| < n$*, são chamados de* ***modos paralelos de operação.***

O conjunto $S(t)$ pode ser escolhido aleatoriamente ou de acordo com alguma regra determinística.

Definição 3.7 *Um estado* $V(t)$ *é dito* ***estável*** *se, e somente se,* $V(t) = sgn(WV(t) - T)$*, ou seja, se não existe mudança no estado da rede, não importa qual seja o modo de operação.*

Definição 3.8 *O conjunto de estados distintos da rede* $\{v_1, \ldots, v_k\}$ *é um* ***ciclo*** *de comprimento* k *se uma sequência de avaliações resulta em uma sequência de estados* $V_1, \ldots, V_k, V_1, \ldots$ *que se repetem infinitamente.*

Exemplo 3.1 *Considere a rede da Figura 3.2, em que* $W_{1,2} = -1$ *e* T *é o vetor nulo. Os estados estáveis da rede são* $(-1, 1)$ *e* $(1, -1)$*.*

De fato,

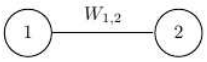


Figura 3.2: Rede neural com 2 nós.

- N é uma rede neural com 2 nós ($n = 2$);
- $W_{1,2} = -1$;
- $T = 0$;
- W é a matriz simétrica $n \times n$ de diagonal zero com $W_{i,j}$ igual ao peso associado a aresta (i, j), ou seja,
$$W_{2\times 2} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix};$$
- Como cada nó pode estar em um dos dois estados possíveis 1 ou -1, suponha que o estado inicial seja $(-1, 1)$. Para $V_1(t) = -1$, o próximo estado do nó é

$$V_1(t+1) = sgn(H_1(t)) = -1, \quad \text{pois} \quad H_i(t) = \sum_{j=1}^{n} W_{ji}V_j(t) - T_i,$$

o que implica

$$H_1(t) = W_{1,1}V_1(t) + W_{2,1}V_2(t) = 0 \cdot (-1) + (-1) \cdot 1 = -1.$$

Para $V_2(t) = 1$, o próximo estado do nó é

$$V_2(t+1) = sgn(H_2(t)) = 1, \ \text{ pois}$$

$$H_2(t) = W_{1,2}V_1(t) + W_{2,2}V_2(t) = (-1) \cdot (-1) + 0 \cdot 1 = 1.$$

Portanto, como o segundo estado da rede é $V(t) = (-1, 1)$, temos que $(-1, 1)$ é um estado estável da rede.

De forma análoga, supondo agora que o estado inicial da rede é $(1, -1)$, obtém-se que $(1, -1)$ é também um estado estável da rede.

Uma das propriedades mais importantes do modelo é a ***propriedade de convergência*** descrita no Teorema 3.1. Esta propriedade é a base para as possíveis aplicações do modelo, como dispositivos de memória associativa (dispositivos que permitem armazenar itens que depois podem ser recuperados baseando-se apenas em informação incompleta sobre o próprio item a ser recuperado)[2] e otimização combinatória[3].

Teorema 3.1 ***(Propriedade de convergência:)*** *[8, Teorema 1] Seja $N = (W, T)$ uma rede neural, com W a matriz simétrica.*

1) Se N opera em um modo serial e os elementos da diagonal de W são não negativos, a rede sempre converge para um estado estável.

[2] Uma *memória* é um dispositivo que permite um computador armazenar dados temporariamente ou permanentemente.

[3] *Otimização combinatória* é um ramo da Ciência da Computação e da Matemática Aplicada que estuda problemas de otimização em conjuntos finitos.

2) Se N está operando em um modo totalmente paralelo a rede sempre converge para um estado estável ou para um ciclo de comprimento no máximo 2.

Demonstração do Teorema 3.1 (1):

Faremos aqui apenas a prova do item (1) do teorema, uma vez que apenas esta nos será necessária no desenvolvimento deste trabalho. A demonstração do item (2) pode ser encontrada em [7].

A ideia principal da prova nas duas partes do teorema é definir uma chamada **função de energia**, limitada superiormente e mostrar que esta função de energia é não decrescente quando o estado da rede muda como resultado do cálculo, o que faz com que tal função seja convergente. O segundo passo na prova é mostrar que a função de energia constante implica, no primeiro caso, um estado estável e, no segundo caso, um ciclo de comprimento igual a, no máximo, 2.

Sejam $N = (W, T)$ uma rede neural operando em modo serial, W a matriz simétrica e defina a *função de energia*

$$E(t) = V^T(t)WV(t) - 2V^T(t)T. \tag{3.2}$$

Seja $\Delta E = E(t+1) - E(t)$ a diferença associada a dois estados da rede consecutivos e ΔV_k a diferença entre o próximo estado e o estado atual do nó k para algum tempo arbitrário t, ou seja,

$$\Delta V_k = V_k(t+1) - V_k(t) = sgn(H_k(t)) - V_k(t).$$

Logo, como cada nó pode estar em um dos dois estados possíveis, 1 ou -1, temos

$$\Delta V_k = \begin{cases} 0 & \text{se } V_k(t) = sgn(H_k(t)) \\ -2 & \text{se } V_k(t) = 1 \ \ e \ \ sgn(H_k(t)) = -1 \\ 2 & \text{se } V_k(t) = -1 \ \ e \ \ sgn(H_k(t)) = 1. \end{cases} \tag{3.3}$$

Como por hipótese a rede está operando em um modo serial, o cálculo (3.1) é realizado em um único nó em qualquer momento determinado. Suponha então que tal cálculo seja realizado em um nó arbitrário k. A diferença de energia resultante deste cálculo é

$$\Delta E = \Delta V_k \left(\sum_{j=1}^{n} W_{k,j} V_j + \sum_{i=1}^{n} W_{i,k} V_i \right) + W_{k,k} \Delta V_k^2 - 2\Delta V_k T_k. \tag{3.4}$$

Pela simetria de W e pela definição $H_k(t) = \sum_{j=1}^{n} W_{j,k} V_j(t) - T_k$, temos

$$\begin{aligned} \Delta E &= \Delta V_k \left(H_k + T_k + H_k + T_k \right) + W_{k,k} \Delta V_k^2 - 2\Delta V_k T_k \\ &= \Delta V_k \left(2H_k + 2T_k \right) + W_{k,k} \Delta V_k^2 - 2\Delta V_k T_k \\ &= 2\Delta V_k H_k + 2\Delta V_k T_k + W_{k,k} \Delta V_k^2 - 2\Delta V_k T_k \end{aligned}$$

Donde segue

$$\Delta E = 2\Delta V_k H_k + W_{k,k} \Delta V_k^2 \tag{3.5}$$

Note agora que devido a (3.1) e (3.3) $\Delta V_k H_k \geq 0$, ainda, $\Delta V_k^2 \geq 0$ e por hipótese também temos $W_{k,k} \geq 0$. Disto, segue que

$$\Delta E = 2\Delta V_k H_k + W_{k,k} \Delta V_k^2 \geq 0$$

o que implica $\Delta E \geq 0$ para todo nó k. Assim, $\Delta E = E(t+1) - E(t) \geq 0$ implica

$E(t) \leq E(t+1)$ e temos E monótona decrescente. Como $E(t)$ é limitada superiormente, obtemos E convergente.

O segundo passo nesta demonstração é mostrar que a convergência da função de energia implica na convergência para um estado estável. Os seguintes fatos são úteis nesta etapa:

1. Se $\Delta V_k = 0$, então $\Delta E = 0$;
2. Se $\Delta V_k \neq 0$, então $\Delta E = 0$ somente se a mudança em V_k é de -1 para 1 como $H_k = 0$.

De fato, em (1), se $\Delta V_k = 0$, então $V_k(t+1) = V_k(t)$, para todo k. Assim, $\Delta E = E(t+1) - E(t) = 0$ e $E(t+1) = E(t)$, logo a rede converge para um estado estável, o que corresponde a um valor de máximo local da função de energia $E(t)$.

Agora, em (2), se $\Delta V_k \neq 0$, então $V_k(t+1) \neq V_k(t)$ e $\Delta E > 0$, ou seja, a energia é não decrescente ($E(t+1) > E(t)$), conforme o estado da rede muda. Porém, como a energia é convergente, existe algum $t \in \mathbb{N}$ tal que $E(t+1) = E(t)$. Note que $\Delta E = 0$, isto é, $E(t+1) = E(t)$ somente se V_k muda de -1 para 1 com $H_k = 0$.

Uma vez que a função de energia tenha convergido, do que foi mostrado acima, podemos concluir que a rede alcançará um estado estável após, no máximo, n^2 intervalos de tempo.

■

O Teorema 3.1 implica que uma rede, operando em um modo serial, sempre chegará a um estado estável que corresponde a um máximo local da função de energia E. Esta propriedade sugere o uso da rede como um dispositivo para realizar um algoritmo de busca local, a fim de encontrar um valor máximo da função de energia E.

O valor da função de energia E que corresponde ao estado inicial é melhorado executando uma seqüência de iterações em série aleatórias até que a rede atinja um máximo local. O algoritmo de busca local realizado pelo modelo de rede neural é imposto pela maneira como a rede está operando.

Exemplo 3.2 *Considere uma rede N operando em modo serial e denote por L_N o* ***algoritmo de busca local*** *realizado pela rede que funciona do seguinte modo.*

1. *Comece com uma atribuição aleatória $V \in \{-1, 1\}^n$.*
2. *Escolha um nó $i \in \{1 \dots n\}$ aleatoriamente.*
3. *Tente melhorar E executando a avaliação*

$$v_i = sgn\left(\sum_{j=1}^{n} w_{j,i} v_j - t_i\right).$$

4. *Volte para o passo 2.*

Cada problema de otimização que pode ser definido na forma de uma função quadrática sobre $\{-1, 1\}^n$, como em (3.2), pode ser mapeada por uma rede neural que irá realizar uma busca por sua situação ótima.

Um dos problemas de otimização, que não é apenas representável por uma função quadrática, mas é equivalente a isto, é o problema de encontrar o *corte mínimo* em um grafo. A relação entre o corte mínimo e o problema de maximizar a função de energia de uma rede neural pode ser resumida pelo seguinte teorema:

Teorema 3.2 *[6, Proposição 2] Seja $N = (W, T)$ uma rede neural com todos os limites sendo zero $(T = 0)$. O problema de encontrar um estado V para o qual a energia E é máxima é equivalente a encontrar um corte mínimo no grafo correspondente.*

Demonstração: Como a função de energia é $E(t) = V^T(t)WV(t) - 2V^T(t)T$ e $T = 0$, temos

$$E = \sum_{i=1}^{n}\sum_{j=1}^{n} W_{i,j}V_iV_j.$$

Denotemos por W^{++} a soma dos pesos das arestas em N com ambos os pontos de extremidade iguais a 1, W^{--} a soma dos pesos das arestas em N com ambos os pontos de extremidade iguais a -1 e por W^{+-} a soma dos pesos das arestas em N com os pontos de extremidade iguais a 1 e -1. Disto, segue

$$E = 2(W^{++} + W^{--} - W^{+-}) \tag{3.6}$$

que, somando $2W^{+-} - 2W^{+-}$ à equação (3.6), também pode ser escrito como

$$E = 2(W^{++} + W^{--} + W^{+-}) - 4W^{+-}. \tag{3.7}$$

Como o primeiro termo de (3.7) é constante segue que a maximização de E é equivalente à minimização de W^{+-}. Note que W^{+-} é o peso do corte em N com $\hat{\mathrm{V}}_1$ sendo o conjunto de nós de N com um estado igual a 1.

■

Exemplo 3.3 *Seja $N = (W, T)$ uma rede neural com todos os limites iguais a zero $(T = 0)$ e considere o grafo não orientado da Figura 3.3, cuja matriz de adjacência é dada por $W_{4\times4}$.*

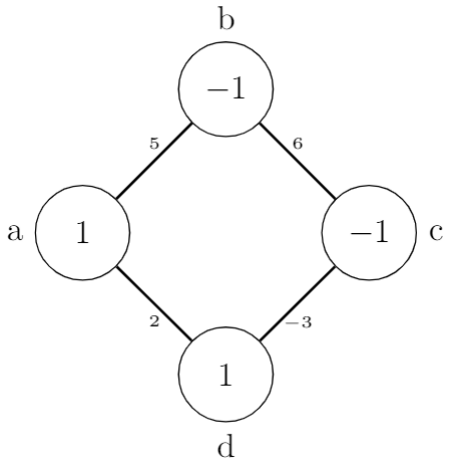


Figura 3.3

$$W_{4\times4} = \begin{pmatrix} 0 & 5 & 0 & 2 \\ 5 & 0 & 6 & 0 \\ 0 & 6 & 0 & -3 \\ 2 & 0 & -3 & 0 \end{pmatrix}.$$

Considerando,
W^{++} *a soma dos pesos das arestas em N com ambos pontos de extremidade iguais a 1,* W^{+-} *a soma dos pesos das arestas em N com os pontos de extremidade iguais a 1 e -1 e* W^{--} *a soma dos pesos das arestas em N com ambos pontos de extremidade iguais a -1, temos:*

Arestas	*Soma*
(a,d)	$W^{++} =$ *2*
(a,b),(d,c)	$W^{+-} =$ *-3 + 5 = 2*
(b,c)	$W^{--} =$ *6*

logo, tomando $W^{+-} = -3 + 5$ *a soma do peso das arestas em N com os pontos de extremidade iguais a* 1 *e* -1*, temos*

$$E = 2(W^{++} + W^{--} + W^{+-}) - 4W^{+-} = 2(2 + 6 + 2) - (4 \cdot 2) = 12.$$

Portanto a função de energia, para este estado da rede, assume valor igual a 12 *e o corte do grafo será dado pelo conjunto de arestas* $\{(a, b), (d, c)\}$ *(sendo* $\hat{V}_1 = \{a, d\}$ *e* $\hat{V}_{-1} = \hat{V} - \hat{V}_1 = \{b, c\}$*).*

Note que a função de energia assumirá valor máximo, igual a 24, no estado da rede $V = (-1, -1, -1, 1)$ *e o corte mínimo do grafo será dado pelo conjunto de arestas* $\{(d, a), (d, c)\}$*, cujo peso é igual a* -1*.*

3.3 Redes Neurais e Códigos Teóricos de Grafos

No artigo [6, p. 979], os autores buscam definir um processo de decodificação por máxima verosimilhança para os códigos teóricos de grafos. Ao se estudar com mais detalhes o artigo, percebe-se que de fato os processos descritos utilizam a procura da palavra do código mais próxima da palavra recebida, em relação à distância de Hamming. Este método é conhecido na literatura como **decodificação pela distância mínima**.

Neste trabalho, no entanto, optamos por manter a linguagem utilizada no artigo estudado.

Na Seção 3.1 mostramos que encontrar o máximo global de uma função de energia é equivalente a encontrar o corte mínimo em um determinado grafo. Nesta seção, em que definiremos os *códigos corretores de erros teóricos de grafos*, o objetivo é provar que a decodificação de máxima verossimilhança em um código teórico de grafo é equivalente a encontrar o corte mínimo do grafo, donde segue, pelo mostrado na Seção 3.1, que a decodificação de máxima verossimilhança em um código teórico de grafo é equivalente a encontrar o máximo da função de energia na rede neural.

Com isto teremos estabelecida uma relação entre redes neurais e códigos corretores de erros teóricos de grafos.

Definição 3.9 *Seja* $\hat{G} = (\hat{V}, \hat{E})$ *um grafo não orientado. Um subconjunto do conjunto de arestas de* $\hat{G}$ *pode ser representado por um* ***vetor característico*** *de comprimento* $|\hat{E}|$*,*

com aresta e_i correspondendo à i-ésima entrada do vetor característico, isto é, cada $S \subseteq \hat{E}$ pode ser representado por um vetor, denotado por 1_s, tal que

$$1_s(i) = \begin{cases} 1, & se\ \ e_i(t) \in S, \\ 0, & caso\ contrário. \end{cases} \tag{3.8}$$

Definição 3.10 *A **matriz de incidência** de um grafo $\hat{G} = (\hat{V}, \hat{E})$, denotada por $D_{\hat{G}}$, que foi definida em (2.11), pode agora ser definida como uma matriz $|\hat{V}| \times |\hat{E}|$ na qual a linha i é o vetor característico do conjunto de arestas incidentes sobre o nó $i \in \hat{V}$.*

Exemplo 3.4 *Seja $\hat{G}$ o grafo de 5 nós e 8 arestas da Figura 3.4.*

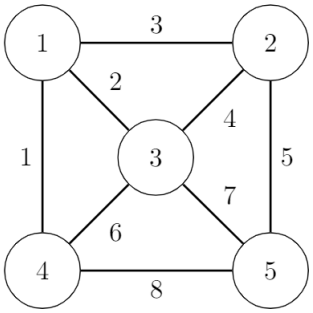


Figura 3.4

O vetor característico de comprimento $|\hat{E}| = 8$ sobre o nó 1 é (1 1 1 0 0 0 0 0) uma vez que as arestas e_1, e_2 e e_3 estão conectadas ao nó 1. Assim, a matriz de incidência do grafo $\hat{G}$ é dada por:

$$D_{\hat{G}} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Existem dois tipos de códigos lineares associados à qualquer grafo linear, a saber os **códigos de circuitos** e os **códigos de corte**, que são chamados de **códigos teóricos de grafos** [38, p. 136].

Definição 3.11 *Um **código de circuito** de um grafo consiste no conjunto das $n-$uplas que correspondem aos circuitos (caminhos fechados) e uniões disjuntas de circuitos. Um **código de corte** de um grafo consiste no conjunto das $n-$uplas que correspondem ao conjunto de cortes e união disjuntas de conjuntos de cortes.*

Neste trabalho discutimos apenas a respeito dos códigos de corte. Os seguintes fatos são as bases para a definição da família dos códigos teóricos de grafos:

- O espaço linha da matriz $D_{\hat{G}}$ é o subespaço de corte de $\hat{G}$.
- O conjunto de vetores característicos que correspondem aos cortes em um grafo conexo $\hat{G} = (\hat{V}, \hat{E})$ formam um espaço vetorial sobre $\mathbb{F}_2$, o espaço linha de $D_{\hat{G}}$, de dimensão $|\hat{V}| - 1$;

- O espaço vetorial que corresponde aos cortes de um grafo $\hat{G}$ é chamado **espaço de corte** de $\hat{G}$.

Definição 3.12 *O espaço linha da matriz de incidência $D_{\hat{G}}$ de um grafo $\hat{G} = (\hat{V}, \hat{E})$ é o subespaço de corte de $\hat{G}$.*

Esta definição pode ser justificada observando que cada linha de $D_{\hat{G}}$ corresponde a um vetor característico de um corte. De fato, seja $\hat{V} = \hat{V}_1 \cup \hat{V}_2$ uma partição do conjunto de vértices $\hat{V}$ em subconjuntos disjuntos e não vazios $\hat{V}_1$ e $\hat{V}_2$. Da definição de corte em um grafo, página 25, sabemos que o conjunto de arestas com uma extremidade em $\hat{V}_1$ e outra em $\hat{V}_2$ determina um corte.

Denotemos este corte por χ e seja y um vetor $m \times 1$ cujas componentes são indexadas por $\hat{E}$, tal que cada coordenada y_i de y é definida da seguinte forma:

$$y_i = \begin{cases} 0 \;\; se \;\; e_i \notin \chi, \\ 1 \;\; se \;\; e_i \in \chi. \end{cases} \tag{3.9}$$

O vetor y é chamado o **vetor de incidência** do corte χ. Definindo agora um vetor u, $n \times 1$, cujas componentes são indexadas por $\hat{V}$, temos

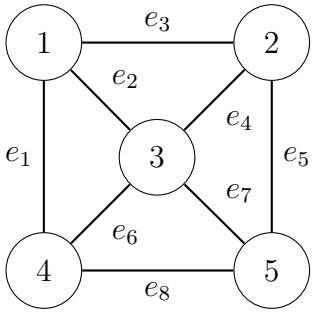
$$u_i = \begin{cases} 0 \;\; se \;\; v_i \notin \hat{V}_1, \\ 1 \;\; se \;\; v_i \in \hat{V}_1. \end{cases} \tag{3.10}$$

Note que, como dois conjuntos de vértices disjuntos vão determinar o mesmo corte, podemos considerar, sem perda de generalidade, $\hat{V}_1$ na definição de u. Destas definições, temos então

$$y^T = u^T \cdot D_{\hat{G}}, \tag{3.11}$$

ou seja, y pertence ao espaço linha de $D_{\hat{G}}$.

Como exemplo, considere o seguinte grafo $\hat{G}$ com 5 vértices e 8 arestas



e seja $\hat{V} = \hat{V}_1 \cup \hat{V}_2$ uma partição de $\hat{G}$ definindo um corte χ tal que $\hat{V}_1 = \{2, 3, 5\}$ e $\hat{V}_2 = \{1, 4\}$. Pela definição temos

$$y^T = (0\ 1\ 1\ 0\ 0\ 1\ 0\ 1)^T \;\text{ e }\; u^T = (0\ 1\ 1\ 0\ 1),$$

e de fato,

$$u^T \cdot D_{\hat{G}} = (0\ 1\ 0\ 0\ 0) \cdot \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} = y^T.$$

Observe que y^T descreve um corte no grafo e este vetor pode ser escrito como uma combinação linear de qualquer 4 linhas de $D_{\hat{G}}$ que formam uma base do espaço de corte, a saber,

$$y^T = 1 \cdot (1\ 1\ 1\ 0\ 0\ 0\ 0\ 0) + 1 \cdot (1\ 0\ 0\ 0\ 0\ 1\ 0\ 1).$$

Note que qualquer outro corte no grafo descrito por um vetor y pode ser também escrito como uma combinação linear das linhas da matriz de incidência do grafo e portanto cada linha de $D_{\hat{G}}$, que é um *vetor característico do conjunto de arestas* incidentes sobre um vértice i em $\hat{V}$, pode ser vista como um *vetor característico de um corte* definido sobre um vértice i em $\hat{V}$.

Por questões práticas, consideramos o conjunto dos cortes por cada vértice e, os vetores característicos obtidos desta maneira nos darão um conjunto gerador "canônico" para o espaço de corte.

Assim, a transformação definida em (3.11) compreende cada conjunto $\hat{V}_1$ e $\hat{V}_2$ como dois vértices e nos dá o conjunto de arestas incidentes sobre eles.

Deste modo, o conjunto de vetores característicos que correspondem aos cortes em um grafo conexo $\hat{G}$ formam um espaço vetorial sobre $\mathbb{F}_2$, o espaço linha de $D_{\hat{G}}$, de dimensão $|\hat{V}| - 1$. Tal espaço vetorial correspondente aos cortes do grafo é chamado **espaço de corte** de $\hat{G}$. A proposição abaixo mostra que a dimensão do espaço é de fato $|\hat{V}| - 1$.

Proposição 3.1 *[4, Lema 2.2] Dado um grafo conexo $\hat{G} = (\hat{V}, \hat{E})$, a matriz de incidência de $\hat{G}$ tem posto $|\hat{V}| - 1$.*

Demonstração: Sejam $\hat{G}$ um grafo conexo e $D_{\hat{G}}$ a matriz de incidência $n \times m$ do grafo, com $n = |\hat{V}|$ e $m = |\hat{E}|$, considere $x^T = (x_1\ x_2\ \ldots\ x_n)^T$ um vetor coluna no espaço nulo de $D_{\hat{G}}^T$, ou seja, $x^T \cdot D_{\hat{G}} = 0_{1\times m}$. Note que $x_i - x_j = 0$ sempre que o vértice v_i estiver conectado ao vértice v_j por meio de algum percurso (uma sequência qualquer de vértices adjacentes) pela construção da matriz de incidência.

Uma vez que $\hat{G} = (\hat{V}, \hat{E})$ é conexo, sempre há uma aresta conectando dois vértices do grafo, desta forma sempre existirá um vértice v_i conectado a um vértice v_j tal que $x_i - x_j = 0$ o que implica $x_i = x_j$ para todo i e j. Assim, todas as coordenadas de x são iguais e podemos escrever $x = \alpha(1\ 1\ \ldots 1)$ para algum $\alpha \in \mathbb{R}$.

Portanto, o espaço nulo de $D_{\hat{G}}^T$ é no máximo unidimensional, donde segue que o posto de $D_{\hat{G}}^T$ é no mínimo $n - 1$. Temos agora dois casos a analisar:

Se $\hat{G}$ é um grafo orientado, pela definição da matriz de incidência 2.1 temos que as somas das entradas de cada coluna de $D_{\hat{G}}^T$ é igual a zero e consequentemente as linhas de $D_{\hat{G}}^T$ são linearmente dependentes donde segue que o posto de $D_{\hat{G}}^T$ é no máximo $n - 1$.

Se por outro lado $\hat{G}$ não é um grafo orientado, como estamos considerando um espaço formado sobre $\mathbb{F}_2$, somando as entradas de cada coluna de $D_{\hat{G}}^T$ obtemos novamente a soma igual a zero e o resultado segue como no caso anterior. Logo $\hat{G}$ tem posto $n - 1 = |\hat{V}| - 1$.

■

Assim, dado um grafo conexo $\hat{G}$, o espaço de corte do grafo é um **código de bloco linear** de dimensão $|\hat{V}| - 1$.

Portanto, cada grafo conexo tem um $\left[|\hat{\mathrm{E}}|\ ,\ (|\hat{\mathrm{V}}| - 1)\right]-$código associado a seus cortes. Este código será denotado por $C_{\hat{G}}$.

Definição 3.13 *Os códigos associados a grafos, os códigos de corte, são chamados de* ***códigos teóricos de grafos.***

Exemplo 3.5 *Seja $\hat{G}$ o grafo do exemplo* (3.4) *cuja matriz de incidência é dada por*

$$D_{\hat{G}} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Quaisquer 4 linhas de $D_{\hat{G}}$ formam uma base do espaço de corte de $\hat{G}$. Por exemplo, as primeiras quatro linhas de $D_{\hat{G}}$ formam uma base do espaço de corte de $\hat{G}$ e portanto, a matriz formada por estas linhas é uma matriz geradora do código de bloco corretor de erros linear associado a $\hat{G}$.

Ainda, $\hat{G}$ não contém um corte com menos de três arestas (além do corte vazio). Assim, o código $C_{\hat{G}}$ tem uma distância de Hamming mínima 3 *e pode corrigir um erro.*

Na Teoria dos Códigos Corretores de Erros, a decodificação é um passo importante para a recuperação da informação transmitida. A pergunta que queremos responder agora é: como formular o problema de decodificação de **máxima verossimilhança (MDL)** do código $C_{\hat{G}}$ em uma linguagem teórica dos grafos?

Em outras palavras, dado um grafo $\hat{G} = (\hat{V}, \hat{E})$ e um vetor $Y \in \{0,1\}^{|\hat{E}|}$, qual a palavra do código em $C_{\hat{G}}$ que mais se aproxima de Y, pela distância de Hamming? Para responder esta questão mostramos inicialmente o seguinte resultado.

Lema 3.1 *[6, Lema 1] Sejam $\hat{G} = (\hat{V}, \hat{E})$ um grafo conexo e $C_{\hat{G}}$ o código associado a $\hat{G}$. Seja Y um vetor em $\{0,1\}^{|\hat{E}|}$. Construímos um novo grafo, que será denotado por $\hat{G}_Y$, atribuindo pesos às arestas de $\hat{G}$ da seguinte maneira*

$$W_i = (-1)^{Y_i},$$

com W_i o peso associado com a aresta i em $\hat{G}$. Então a MDL de Y com respeito a $C_{\hat{G}}$ é equivalente a encontrar o corte mínimo em $\hat{G}_Y$.

Demonstração: Suponha que o número de uns em Y seja a. Seja M uma palavra do código arbitrária em $C_{\hat{G}}$ e denote por $N^{i,j}$ o número de posições em que M contém um $i \in \{0,1\}$ e Y contém um $j \in \{0,1\}$. Assim, $a = N^{0,1} + N^{1,1}$ e daí

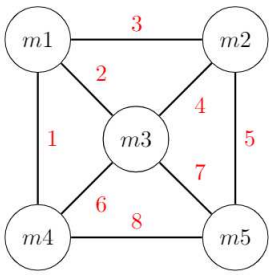
$$-N^{1,1} + N^{1,0} = N^{0,1} - a + N^{1,0}. \tag{3.12}$$

Portanto, minimizar o lado direito de (3.12) sobre todo $M \in C_{\hat{G}}$ é equivalente a encontrar uma palavra do código que seja mais próxima de Y. Por outro lado, minimizar o lado esquerdo é equivalente a encontrar o corte mínimo em $\hat{G}_Y$.

■

Para ilustrar o lema e com o intuito de tornar mais clara a explicação feita a respeito da equação (3.12), observe o seguinte exemplo.

Exemplo 3.6 *Sejam $\hat{G}$ o grafo da Figura 3.5, $D_{\hat{G}}$ sua matriz de incidência e $G_{\hat{G}}$ a matriz geradora do código $C_{\hat{G}}$ associado aos cortes do grafo.*



$$D_{\hat{G}} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

$$G_{\hat{G}} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Figura 3.5

Considere $Y = (0\ 0\ 1\ 1\ 0\ 1\ 1\ 1)$ um vetor em $\{0,1\}^{|\hat{E}|}$. Na equação (3.12), note que minimizar o lado direito, isto é, minimizar

$$N^{0,1} - a + N^{1,0}$$

sobre todo $M \in C_{\hat{G}}$ é equivalente a encontrar uma palavra do código que seja mais próxima do vetor Y, pois $N^{0,1} - a + N^{1,0}$ é igual a $-N^{1,1} + N^{1,0}$ em que

$N^{1,0}$ = *número de posições em que M contém 1 e Y contém 0 e*

$N^{1,1}$ = *número de posições em que M contém 1 e Y contém 1.*

Desta forma, encontramos o número de entradas de M que são diferentes das entradas de Y, logo pela distância de Hamming, encontramos a palavra M mais próxima, com a menor distância, de Y.

Por outro lado, minimizar o lado esquerdo da equação (3.12), isto é, minimizar

$$-N^{1,1} + N^{1,0}$$

sobre todo $M \in C_{\hat{G}}$ é equivalente a encontrar o corte mínimo em $\hat{G}_Y$, pois calculando $N^{0,1} - a + N^{1,0}$ em que

$N^{0,1}$ = *número de posições em que M contém 0 e Y contém 1 e*

$N^{1,0}$ = *número de posições em que M contém 1 e Y contém 0*

encontramos a palavra mais distante possível de Y. Como cada linha i da matriz incidente é um vetor característico do conjunto de arestas sobre o nó $i \in \hat{V}$, temos que cada entrada deste vetor relaciona o nó com uma aresta e, como o peso de cada aresta é calculado por $W_i = (-1)^{Y_i}$, o número $N^{0,1} - a + N^{1,0}$, com a o número de uns em Y, associa a cada nó o peso do corte considerando suas arestas incidentes.

De fato, vamos minimizar

$$-N^{1,1} + N^{1,0}$$

sobre todo $M \in C_{\hat{G}}$. Construindo a seguinte tabela

	$-N^{1,1}$	$N^{1,0}$	$-N^{1,1}+N^{1,0}$
m_1	*1*	*2*	*1*
m_2	*2*	*1*	*-1*
m_3	*3*	*1*	*-2*
m_4	*2*	*1*	*-1*

podemos observar que m_3 é a palavra mais próxima de Y (enquanto m_1 é a mais distante). Assim, olhando para $m_3 = (0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ )$ [4] temos que as arestas e_2, e_4, e_6 e e_7 estão conectadas ao nó m_3. Daí, calculando o peso associado a cada aresta, temos

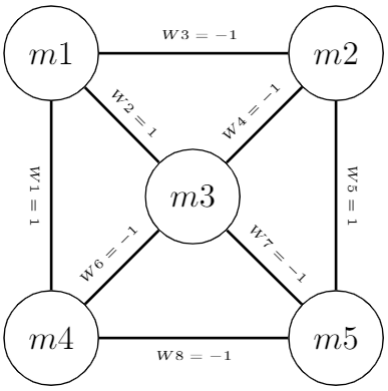
$$W_2 = 1, W_4 = -1, W_7 = -1 \ \ e \ \ W_6 = -1$$

e o peso do corte, que é a soma do peso destas arestas, é igual a -2. Portanto, quando minimizamos $N^{0,1} + N^{1,0}$, tomamos a palavra mais próxima de Y, e ao calcular $N^{0,1} + N^{1,0} - a$, tomamos o número de entradas diferentes e subtraímos do valor encontrado o número de uns em Y que faz com que o peso de cada aresta seja igual a -1. Desta forma, ao encontrar a palavra mais próxima de Y, encontramos o corte mínimo em $\hat{G}_Y$.

Por fim, construimos o novo grafo atribuindo pesos às arestas de $\hat{G}$, com

$$\begin{array}{lll} W_1 = (-1)^0 = 1 & W_2 = (-1)^0 = 1 & W_3 = (-1)^1 = -1 \\ W_4 = (-1)^1 = -1 & W_5 = (-1)^0 = 1 & W_6 = (-1)^1 = -1 \\ W_7 = (-1)^1 = -1 & W_8 = (-1)^1 = -1, & \end{array}$$

temos



Teorema 3.3 *[6, Teorema 1] Seja $\hat{G} = (\hat{V}, \hat{E})$ um grafo conexo. Então a MDL de uma palavra de Y com respeito a $C_{\hat{G}}$ é equivalente a encontrar o máximo da função de energia E da rede neural definida pelo grafo $\hat{G}_Y$ com todos os seus valores limites iguais a zero.*

Demonstração: Pelo Lema 3.1, a MDL de Y com respeito a $C_{\hat{G}}$ é equivalente a encontrar o corte mínimo em $\hat{G}_Y$. Pelo Teorema 3.2, encontrar o corte mínimo em um grafo é equivalente a encontrar o máximo da função de energia de uma rede neural definida por um grafo com todos os limites sendo zero.

■

[4] Aqui estamos fazendo um abuso de notação, considerando cada nó como uma linha de $D_{\hat{G}}$.

Distância mínima

A distância mínima de um código teórico de grafo é dado pelo número de arestas no conjunto de corte que possuí o menor número de arestas [38].

Para ter distância d, cada nó em um código do conjunto de corte deve ter grau maior ou igual a d, isto é, deve possuir pelo menos d arestas incidentes sobre ele.

Como uma aresta é incidente em dois nós, para os códigos corretores de erros teóricos de grafos temos

$$d^* \leqslant \frac{2 \cdot |\hat{\mathrm{E}}|}{|\hat{\mathrm{V}}|},$$

com d^* a distância mínima do código.

De fato, como cada linha i da matriz $D_{\hat{G}}$ é um vetor característico do conjunto de arestas incidentes sobre um vértice $i \in \hat{V}$, pela Proposição 2.1, considerando $|\hat{V}|$ o número de vértices do grafo e $|\hat{E}|$ o número de arestas, temos

$$|\hat{E}| = \left(\frac{g(v_1) + g(v_2) + \ldots + g(v_n)}{2} \right)$$

como cada nó em um código de corte deve ter grau $\geq d^*$, logo

$$|\hat{E}| = \left(\frac{g(v_1) + g(v_2) + \ldots + g(v_n)}{2} \right) \geq \frac{d^* + \ldots + d^*}{2} \geq \frac{|\hat{V}| d^*}{2} \Rightarrow d^* \leq \frac{2|\hat{E}|}{|\hat{V}|}.$$

Exemplo 3.7 *Um* $[7,4]-$*código de Hamming não é um código teórico de grafo, pois tem distância mínima* 3 *e* $\dfrac{14}{5} < 3$.

3.4 Códigos corretores de erros e funções de energia

Nesta seção estendemos os resultados da Seção 3.3 e mostramos que encontrar o máximo global de uma função de energia definida com base na matriz geradora de um dado código de bloco linear é equivalente à decodificação de máxima verossimilhança do código.

Considere um $[n,k]$-código corretor de erros linear $\mathscr{C}$ definido por uma $k \times n$ matriz geradora G. Um vetor de informação $b = (b_1, b_2, \ldots, b_k)$ é codificado na palavra código $v = (v_1, v_2, \ldots, v_n)$ por

$$\begin{array}{ccc} \mathbb{F}^k & \longrightarrow & \mathbb{F}^n \\ (b_1, b_2, \ldots, b_k) & \longmapsto & (b_1, b_2, \ldots, b_k) \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{pmatrix} = (v_1, v_2, \ldots, v_n). \end{array}$$

Deste modo, dado o vetor $b = (b_1, b_2, \ldots, b_k)$, obtém-se:

$$v_1 = b_1 g_{11} + b_2 g_{21} + b_3 g_{31} + \ldots + b_k g_{k1}$$

$$v_2 = b_1g_{12} + b_2g_{22} + b_3g_{32} + \ldots + b_kg_{k2}$$

$$\vdots$$

$$v_n = b_1g_{1n} + b_2g_{2n} + b_3g_{3n} + \ldots + b_kg_{kn}.$$

Para estendermos os resultados da Seção 3.3, faremos uma **mudança de alfabeto**. A ideia é representar os símbolos do grupo aditivo $\mathbb{Z}_2$ como símbolos no grupo multiplicativo $\{1, -1\}$ pela transformação

$$\begin{array}{ccc} \mathbb{Z}_2 & \longrightarrow & \{1,-1\} \\ a & \longmapsto & (-1)^a, \end{array}$$

isto é,

$$0 \longmapsto 1, \quad 1 \longmapsto -1.$$

A partir desta mudança de alfabeto, um vetor informação $b = (b_1, b_2, \ldots, b_k)$ será representado por

$$x = (x_1, x_2, \ldots, x_k), \text{ com } x_i = (-1)^{b_i}$$

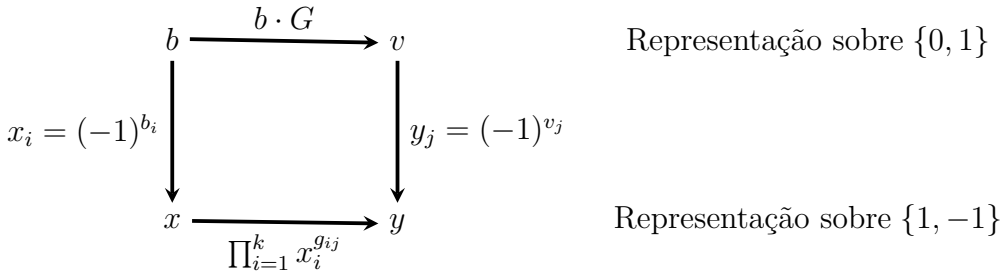
e a codificação da palavra do código $v = (v_1, v_2, \ldots, v_n)$ será representada por $y = (y_1, y_2, \ldots, y_n)$, de modo que

$$y_j = (-1)^{v_j} = (-1)^{\bigoplus_{i=1}^{k} b_i g_{i,j}} = \prod_{i=1}^{k} (-1)^{b_i g_{i,j}} = \prod_{i=1}^{k} x_i^{g_{i,j}}. \tag{3.13}$$

De fato,

$$\begin{array}{rcl} y_j = (-1)^{v_j} & = & (-1)^{\bigoplus_{i=1}^{k} b_i g_{i,j}} \\ & = & (-1)^{b_1g_{1j} + b_2g_{2j} + \ldots + b_kg_{kj}} \\ & = & (-1)^{b_1g_{1j}} \cdot (-1)^{b_2g_{2j}} \cdot \ldots \cdot (-1)^{b_kg_{kj}} \\ & = & \displaystyle\prod_{i=1}^{k} (-1)^{b_i g_{i,j}} = \prod_{i=1}^{k} ((-1)^{b_1})^{g_{i,j}} = \prod_{i=1}^{k} x_i^{g_{i,j}}. \end{array}$$

O seguinte esquema será bastante útil para auxiliar na compreensão dos exemplos subsequentes:



Note que as setas verticais indicam que o vetor de saída é representado pelo vetor de chegada segundo a mudança indicada. Assim, o vetor b é representado pelo vetor x e o vetor v é representado pelo vetor y.

As setas horizontais indicam a codificação, ou seja, o vetor b é codificado como um vetor v na representação sobre $\{0, 1\}$ e o vetor x é codificado como um vetor y na representação sobre $\{1, -1\}$.

Exemplo 3.8 *Considere o* $[7,4]-$*código de Hamming cuja matriz geradora é dada por*

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

Dados os quatro símbolos de informação (b_1, b_2, b_3, b_4), a palavra do código correspondente é $v = (b_1, b_2, b_3, b_4, (b_2 + b_3 + b_4), (b_1 + b_3 + b_4), (b_1 + b_2 + b_4))$.

Na representação sobre $\{1, -1\}$, com $x_j = (-1)^{b_j}$, temos

$$y = (x_1, x_2, x_3, x_4, (x_2x_3x_4), (x_1x_3x_4), (x_1x_2x_4)).$$

De fato, $y_j = (-1)^{v_j} = \prod_{i=1}^{4} x_i^{g_{i,j}}$ nos dá:

$$\begin{aligned}
y_1 &= x_1^{g_{11}} \cdot x_2^{g_{21}} \cdot x_3^{g_{31}} \cdot x_4^{g_{41}} = x_1^1 \cdot x_2^0 \cdot x_3^0 \cdot x_4^0 = x_1, \\
y_2 &= x_1^{g_{12}} \cdot x_2^{g_{22}} \cdot x_3^{g_{32}} \cdot x_4^{g_{42}} = x_1^0 \cdot x_2^1 \cdot x_3^0 \cdot x_4^0 = x_2, \\
y_3 &= x_1^{g_{13}} \cdot x_2^{g_{23}} \cdot x_3^{g_{33}} \cdot x_4^{g_{43}} = x_1^0 \cdot x_2^0 \cdot x_3^1 \cdot x_4^0 = x_3, \\
y_4 &= x_1^{g_{14}} \cdot x_2^{g_{24}} \cdot x_3^{g_{34}} \cdot x_4^{g_{44}} = x_1^0 \cdot x_2^0 \cdot x_3^0 \cdot x_4^1 = x_4, \\
y_5 &= x_1^{g_{15}} \cdot x_2^{g_{25}} \cdot x_3^{g_{35}} \cdot x_4^{g_{45}} = x_1^0 \cdot x_2^1 \cdot x_3^1 \cdot x_4^1 = x_2x_3x_4, \\
y_6 &= x_1^{g_{16}} \cdot x_2^{g_{26}} \cdot x_3^{g_{36}} \cdot x_4^{g_{46}} = x_1^1 \cdot x_2^0 \cdot x_3^1 \cdot x_4^1 = x_1x_3x_4, \\
y_7 &= x_1^{g_{17}} \cdot x_2^{g_{27}} \cdot x_3^{g_{37}} \cdot x_4^{g_{47}} = x_1^1 \cdot x_2^1 \cdot x_3^0 \cdot x_4^1 = x_1x_2x_4.
\end{aligned}$$

Para representar um código linear no alfabeto $\{1, -1\}$, ao invés de utilizarmos uma matriz geradora, utilizaremos um **procedimento de codificação** denotado por $(x \longrightarrow y)$ que associa a cada vetor informação $x = (x_1, x_2, \cdots, x_k)$, outro vetor $y = (y_1, y_2, \ldots, y_n)$ tal que cada y_j é um monômio expresso como produto dos $x_i, i = 1, \ldots, k$.

Definição 3.14 *Um procedimento de codificação é* ***sistemático*** *se, e somente se,* $y_j = x_j$*, para todos* $1 \leq j \leq k$*.*

Exemplo 3.9 *No Exemplo 3.8, observemos que o* $[7,4]$*-código de Hamming é descrito por um procedimento de codificação sistemático, pois* $y_i = x_i$*, para* $i \in \{1, 2, 3, 4\}$*.*

Exemplo 3.10 *O* $R(1,3)$ *código de Reed-Muller* C' *de primeira ordem encurtado*[5] *é descrito pelo processo sistemático de codificação.*

$$(x_1, x_2, x_3) \to (x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1x_2x_3) \tag{3.14}$$

Enquanto o R(1,3) código de Reed-Muller C *de primeira ordem é descrito pelo procedimento de codificação*

$$(x_0, x_1, x_2, x_3) \to (x_0, x_0x_1, x_0x_2, x_0x_1x_2, x_0x_3, x_0x_1x_3, x_0x_1x_3, x_0x_1x_2x_3). \tag{3.15}$$

[5]Veja 1.1.4 no Capítulo 1 para a definição destes códigos.

que não é sistemático.

De fato, como

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{3.16}$$

é uma matriz geradora do código $R(1,3)$, obtém-se

$$G' = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}, \tag{3.17}$$

a matriz geradora do código $R(1,3)$ encurtado C'. Assim, se $b = (b_1, b_2, b_3)$ é um vetor de informação, uma palavra v do código $R(1,3)$ encurtado C' é da forma

$$v = b \cdot G' = (b_1, b_2, b_3, b_1 + b_2, b_1 + b_3, b_2 + b_3, b_1 + b_2 + b_3)$$

e daí obtemos a codificação descrita em (3.14). Por exemplo, tomando $v = (1\ 0\ 1\ 0\ 1\ 0\ 1)$, obtemos

$$b = (1\ 0\ 1), \quad x = (-1, 1, -1) \text{ e } y = (-1, 1, -1, 1, -1, 1, -1)$$

e, portanto, $y_j = x_j$, para todo $1 \leq j \leq 3$.

Por outro lado, tomando $v = (0\ 1\ 0\ 1\ 0\ 1\ 0\ 1)$ uma palavra do $R(1,3)$-código de Reed-Muller de primeira ordem, fazendo $(b_0, b_1, b_2, b_3) \cdot G = v$, obtemos $b = (0\ 1\ 0\ 0)$, logo

$$x = (1, -1, 1, 1) \text{ e } y = (1, -1, 1, -1, 1, -1, 1, -1)$$

e, assim, o este código C não é descrito pelo procedimento sistemático de codificação uma vez que neste exemplo $x_j \neq y_j$, para $j = 4$.

Definição 3.15 *Seja $G = (g_{ij})$ uma matriz $k \times n$ de uns e zeros. A* ***representação polinomial*** *de G com respeito a um vetor $\omega \in \{1, -1\}^n$, denotada por E_ω, é*

$$E_\omega(x) = \sum_{j=1}^{n} w_j \prod_{i=1}^{k} x_i^{g_{i,j}}, \tag{3.18}$$

para todo $x = (x_1, x_2, \ldots, x_k) \in \{1, -1\}^k$.

Considere o código linear definido pela matriz geradora G (ou equivalentemente, pelo procedimento de codificação associado a G). A representação polinomial de G, isto é, E_ω, pode ser chamada a **função de energia** de ω com respeito ao procedimento de codificação $x \longrightarrow y$. Note que $E_\omega(x) = \omega \cdot y(x)$, em que o ponto centrado indica o produto interno canônico em $\mathbb{F}_2^n$.

Para estabelecer a conexão entre função de energia e códigos lineares, provaremos que encontrar o máximo global de E_ω é equivalente à MLD de um vetor ω com respeito ao código C.

Teorema 3.4 *[6, Teorema 2] Dado um (n,k)-código C definido por um procedimento de codificação $x \longrightarrow y$ e um vetor $\omega \in \{1,-1\}^n$, a palavra do código mais próxima (na distância de Hamming) de ω em C corresponde a um vetor de informação $b = (b_1, b_2, \cdots, b_k)$ se, e somente se,*

$$E_\omega(b) = \max_{x \ \in \ \{1,-1\}^k} E_\omega(x).$$

Demonstração: Para qualquer $x \ \in \ \{1,-1\}^k$, temos

$$\begin{aligned} E_\omega(x) &= \sum_{j=1}^{n} \omega_j y_j(x) \\ &= |\{j : \omega_j = y_j(x)\}| - |\{j : \omega_j \neq y_j(x)\}| \\ &= n - 2|\{j : \omega_j \neq y_j(x)\}| \\ &= n - 2d_H(\omega, y), \end{aligned}$$

em que d_H denota a distância de Hamming. A segunda igualdade se obtém observando que nos produtos ocorre o seguinte: ou

i) $w_j = y_j(x)$, com $1 \leq j \leq n$, o que implica $w_j \cdot y_j(x) = 1$ ou

ii) $w_j \neq y_j(x)$, com $1 \leq j \leq n$, o que implica $w_j \cdot y_j(x) = -1$.

Assim, $E_w(x) = w_1 y_1(x) + \ldots + w_n y_n(x) = |\{j : \omega_j = y_j(x)\}| - |\{j : \omega_j \neq y_j(x)\}|$. Ainda, fazendo

$$|\{j : \omega_j = y_j(x)\}| - |\{j : \omega_j \neq y_j(x)\}| = a - b$$

podemos escrever $a = n - b$. Logo, $a - b = n - b - b = n - 2b$ e obtemos assim a terceira igualdade. A expressão $E_w(x) = n - 2d_H(\omega, y)$ implica que $E_w(b_1, b_2, \cdots, b_k)$ pode alcançar um máximo se, e somente se, $d_H(\omega, y)$ atinge um mínimo.

■

Exemplo 3.11 *Considere o $[7,4]$-código de Hamming, definido pelo procedimento de codificação em* (3.19).

$$(x_1, x_2, x_3, x_4) \to (x_1, x_2, x_3, x_4, x_2x_3x_4, x_1x_3x_4, x_1x_2x_4) \tag{3.19}$$

Suponha que queremos executar o MLD da palavra recebida

$$\omega = (1, -1, -1, 1, 1, -1, 1).$$

Então

$$E_\omega(x_1, x_2, x_3, x_4) = \sum_{j=1}^{7} w_j \prod_{i=1}^{4} x_i^{g_{i,j}} = w_1(x_1^{g_{1,1}} x_2^{g_{2,1}} x_3^{g_{3,1}} x_4^{g_{4,1}}) + \ldots + w_7(x_1^{g_{1,7}} x_2^{g_{2,7}} x_3^{g_{3,7}} x_4^{g_{4,7}})$$

$$\Rightarrow E_\omega(x_1, x_2, x_3, x_4) = x_1 - x_2 - x_3 + x_4 + x_2x_3x_4 - x_1x_3x_4 + x_1x_2x_4.$$

O máximo desse polinômio ocorre em $x = (1, -1, -1, 1)$, pois $E_\omega(1, -1, -1, 1) = 5$ é o maior valor que a função pode assumir. Assim, a palavra recebida é decodificada como $(1, -1, -1, 1)$.

Exemplo 3.12 *Considere o $R(1,3)$-código de Reed-Muller de primeira ordem definido pelo processo de codificação em* (3.15)*. Dada a palavra recebida $\omega = (\omega_0, \omega_1, \ldots, \omega_7)$, a função de energia é*

$$\begin{aligned} E_\omega(x_0, x_1, x_2, x_3) &= x_0(\omega_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_1 x_2 + \omega_4 x_3 + \omega_5 x_1 x_3 + \\ &+ \omega_6 x_2 x_3 + \omega_7 x_1 x_2 x_3) \\ &= x_0(\omega_0 + E_{\tilde{\omega}}(x_1, x_2, x_3)), \end{aligned}$$

em que, $E_{\tilde{\omega}}(x_1, x_2, x_3) = \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_1 x_2 + \omega_4 x_3 + \omega_5 x_1 x_3 + \omega_6 x_2 x_3 + \omega_7 x_1 x_2 x_3$ e $\tilde{\omega} = (\omega_1, \ldots, \omega_7)$. Por isso, é suficiente encontrar

$$\max_{(x_1,x_2,x_3) \in \{1,-1\}^3} |E_{\tilde{\omega}}(x_1, x_2, x_3)|$$

Se a energia que corresponde ao máximo é positiva, então $x_0 = 1$. Assuma que recebemos $\omega = (-1, 1, 1-1, 1, 1, 1, -1)$, então

$$E_{\tilde{\omega}}(x_1, x_2, x_3) = x_1 + x_2 - x_1 x_2 + x_3 + x_1 x_3 + x_2 x_3 - x_1 x_2 x_3.$$

Logo

$$\max_{(x_1,x_2,x_3) \in \{1,-1\}^3} |E_{\tilde{\omega}}(x_1, x_2, x_3)| = E_{\tilde{\omega}}(1, 1, 1) = 3.$$

Já que a energia é positiva, a palavra recebida é decodificada como $(1, 1, 1, 1)$*. Neste caso, a decodificação não é única, visto que o máximo é atingido em mais que um ponto, a saber,* $(1, 1, -1, 1)$.

A não unicidade decorre do fato do procedimento de codificação ser não sistemático.

Dado um procedimento de codificação, podemos usar o mesmo argumento que no Teorema 3.4 para expressar a distância miníma de um código C. Considere o procedimento de codificação

$$x = (x_1, x_2, \cdots, x_k) \longrightarrow y = (y_1, y_2, \cdots, y_n)$$

e a função de energia[6] com $\omega = (1, 1, \cdots, 1)$, isto é,

$$E_w(x_1, x_2, \cdots, x_k) = (y_1 + y_2 + \cdots + y_n).$$

Como antes,

$$E_w(x_1, x_2, \cdots, x_k) = n - 2d_h((11\cdots1), (y_1, y_2, \cdots, y_n))$$

e

$$\min_{(x_1,x_2,\cdots,x_k) \neq (11\cdots1)} d_h((11\cdots1), (y_1, y_2, \cdots, y_n))$$

ocorre em

$$M := \max_{(x_1,x_2,\cdots,x_k) \neq (11\cdots1)} E_w(x_1, x_2, \cdots, x_k).$$

Conclui-se, portanto, que d^* (a distância miníma do código) é dada por

$$d^* = \frac{n - M}{2}. \tag{3.20}$$

[6]Veja observação 3.1 ao final da seção.

Exemplo 3.13 *Para o* $[7,4]$ *código de Hamming,*

$$M = \max_{(x_1,x_2,x_3,x_4) \neq (1111)} x_1 + x_2 + x_3 + x_4 + x_2x_3x_4 + x_1x_3x_4 + x_1x_2x_4 = 1.$$

Assim,

$$d^* = \frac{n-M}{2} = \frac{7-1}{2} = 3.$$

Exemplo 3.14 *Para o* $R(1,3)$*-código de Reed-Muller de primeira ordem*

$$E_\omega(x) = \sum_{j=0}^{7} w_j \prod_{i=0}^{3} x_i^{g_{i,j}} = w_0(x_0^{g_{00}} x_1^{g_{10}} x_2^{g_{20}} x_3^{g_{30}}) + \ldots + w_7(x_0^{g_{07}} x_1^{g_{17}} x_2^{g_{27}} x_3^{g_{37}})$$

Olhando para a matriz (3.16) *de* $R(1,3)$*, temos*

$$E_\omega(x) = w_0x_0 + w_1x_0x_1 + w_2x_0x_2 + w_3x_0x_1x_2 + w_4x_0x_3 + w_5x_0x_1x_3 + w_6x_0x_2x_3 + w_7x_0x_1x_2x_3$$

e, como $w = (1\ 1\ 1\ \ldots 1)$*, então*

$$E_\omega(x) = x_0 + x_0x_1 + x_0x_2 + x_0x_1x_2 + x_0x_3 + x_0x_1x_3 + x_0x_2x_3 + x_0x_1x_2x_3.$$

Fazendo

$$M = \max_{(x_0,x_1,x_2,x_3) \neq (1111)} x_0(1 + x_1 + x_2 + x_1x_2 + x_3 + x_1x_3 + x_2x_3 + x_1x_2x_3),$$

podemos escrever

$$M = \max_{(x_0,x_1,x_2,x_3) \neq (1111)} x_0(1 + x_1)(1 + x_2)(1 + x_3). \tag{3.21}$$

O máximo em (3.21) *é* $M = 0$*, pois pelo menos um dos* x_i*, para* $i > 1$*, deve ser igual a* -1*. Assim,*

$$d^* = \frac{n-M}{2} = \frac{8}{2} = 4.$$

Note que o mesmo argumento pode ser usado para qualquer $R(1,m)$*-código obtendo assim*

$$d^* = \frac{2^m}{2} = 2^{m-1}.$$

Observação 3.1 *Faz sentido considerarmos a função de energia com* $\omega = (1\ 1\ \ldots 1) \in \{1,-1\}^n$*, pois* $\omega = (1\ 1\ \ldots 1) \in \{1,-1\}^n$ *corresponde ao vetor* $0 = (0\ 0\ \ldots 0)$*. Como o código* C *é linear, encontrar a distância mínima de* C *é o mesmo que encontrar o peso mínimo de* C*, que na Teoria Clássica é o mesmo que encontrar a palavra mais próxima ao vetor nulo.*

3.5 Representando códigos lineares como estados estáveis de funções de energia

Nesta seção associamos a função de energia com a matriz teste de paridade de um código. Mostramos que cada palavra do código corresponde a um máximo local de um

polinômio associado à esta matriz. Provamos também a implicação contrária, ou seja, que cada máximo local corresponde a uma palavra do código.

Seja C um código linear sobre $\mathbb{F}_2$ definido pela matriz geradora G. Considere a seguinte questão: existe um algoritmo eficiente para construir um polinômio E_C sobre $\{1, -1\}$ (função de energia) com a propriedade que cada máximo local em E_C corresponde a uma palavra no código C e cada palavra em C corresponde a um máximo local em E_C?

A resposta é sim! Nesta seção descrevemos o desenvolvimento de tal algoritmo.

Considere um $[n, k]$-código linear C. Sem perda de generalidade, assuma que a matriz geradora G é dada na forma padrão, isto é,

$$G = [I_k : P], \tag{3.22}$$

com I_k a matriz identidade $k \times k$ e P a matriz $k \times (n-k)$. A transposta da matriz teste de paridade de C é

$$H^T = \begin{bmatrix} P \\ I_{n-k} \end{bmatrix} \tag{3.23}$$

Pela definição de H, para todo $v \in C$, temos

$$vH^T = \overrightarrow{0}, \tag{3.24}$$

Com $\overrightarrow{0}$ o vetor nulo de comprimento $(n-k)$. Assim como em (3.18), escrevemos a representação polinomial da matriz H do seguinte modo, em relação ao vetor de coeficientes com todas as entradas iguais a um. De fato,

$$\tilde{E}_w(x) = \sum_{j=1}^{n-k} w_j \prod_{i=1}^{n} x_i^{h_{ij}} \Rightarrow$$

$$\begin{aligned} \tilde{E}_w(x) &= w_1(x_1^{h_{11}} x_2^{h_{21}} \dots x_n^{h_{n1}}) + w_2(x_1^{h_{12}} x_2^{h_{22}} \dots x_n^{h_{n2}}) + \dots + w_{n-k}(x_1^{h_{1n-k}} x_2^{h_{2n-k}} \dots x_n^{h_{nn-k}}) \\ &= (x_1^{h_{11}} x_2^{h_{21}} \dots x_n^{h_{n1}}) + (x_1^{h_{12}} x_2^{h_{22}} \dots x_n^{h_{n2}}) + \dots + (x_1^{h_{1n-k}} x_2^{h_{2n-k}} \dots x_n^{h_{nn-k}}). \end{aligned} \tag{3.25}$$

Lema 3.2 *[6, Lema 2] Seja $\tilde{E}_\omega(x)$ a representação polinomial de H^T com respeito ao vetor cujas entradas são todas iguais a um como em* (3.25)*. Então $x \in C$ se, e somente se, $\tilde{E}(x) = n - k$.*

Demonstração: ($\Rightarrow$) Suponha $x \notin C$ logo $v \cdot H^T \neq \overrightarrow{0}$ se, e somente se,

$$(x_1, x_2, \dots, x_k, x_{k+1}, x_{k+2}, \dots, x_n) \begin{pmatrix} h_{11} & h_{12} & \dots & h_{1k} & \dots & h_{1(n-k)} \\ h_{21} & h_{22} & \dots & h_{2k} & \dots & h_{2(n-k)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ h_{k1} & h_{k2} & \dots & h_{kk} & \dots & h_{k(n-k)} \\ 1 & 0 & \dots & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \dots & 1 \end{pmatrix} = \tag{3.26}$$

$$= \begin{pmatrix} x_1h_{11} + x_2h_{21} + \ldots + x_kh_{k1} + x_{k+1}1 + x_{k+2}0 + \ldots + x_n0 \\ x_1h_{11} + x_2h_{21} + \ldots + x_kh_{k1} + x_{k+1}0 + x_{k+2}1 + \ldots + x_n0 \\ \vdots \\ x_1h_{1k} + x_2h_{2k} + \ldots + x_kh_{kk} + x_{k+1}0 + x_{k+2}0 + \ldots + x_n0 \\ \vdots \\ x_1h_{1(n-k)} + x_2h_{2(n-k)} + \ldots + x_kh_{k(n-k)} + x_{k+1}0 + x_{k+2}0 + \ldots + x_n1 \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Assim, para alguma linha i, o termo $x_1h_{1i} + x_2h_{2i} + \ldots + x_nh_{nn-k}$, com $h_{ij} \in \{0, 1\}$ e $i \in \{1, \ldots, n\}$, é diferente de zero. Logo existe algum termo x_ih_{ij} igual a 1 (note que aqui tomamos x como uma palavra do código C, ou seja, $x \in \{0, 1\}$. Porém, utilizaremos a mesma letra para representar a mudança de alfabeto na expressão (3.27) abaixo). Desta forma, como

$$\tilde{E}_w(x) = (x_1^{h_{11}}x_2^{h_{21}} \ldots x_k^{h_{k1}}x_{k+1}^1x_{k+1}^0 \ldots x_n^0) + (x_1^{h_{12}}x_2^{h_{22}} \ldots x_k^{h_{kk}}x_{k+1}^0x_{k+2}^1 \ldots x_n^0) +$$

$$+ \ldots + (x_1^{h_{1n-k}}x_2^{h_{2n-k}} \ldots x_k^{h_{k(n-k)}}x_{k+1}^0x_{k+2}^0 \ldots x_n^1),$$

logo, se

$$\tilde{E}_w(x) = (x_1^{h_{11}}x_2^{h_{21}} \ldots x_k^{h_{k1}}x_{k+1}) + (x_1^{h_{12}}x_2^{h_{22}} \ldots x_k^{h_{kk}}x_{k+2}) + \ldots + (x_1^{h_{1n-k}}x_2^{h_{2n-k}} \ldots x_k^{h_{k(n-k)}}x_n) \tag{3.27}$$

é a representação polinomial de H^T, pela mudança de alfabeto, então existe algum $x_i^{h_{ij}} = -1$, donde segue que $\tilde{E}_w(x)$ não possui $n-k$ termos não nulos, isto é, $\tilde{E}_w(x) \neq n-k$.

($\Leftarrow$) Pela representação polinomial de H^T (3.27) com respeito ao vetor cujas entradas são todas iguais a um, segue que $\tilde{E}_w(x)$ tem $n-k$ termos somados. Logo, $\tilde{E}_w(x) = n-k$ se, e somente se, cada termo em $\tilde{E}_w(x)$ é igual a 1. Com isto, se $(x_1^{h_{11}}x_2^{h_{21}} \ldots x_n^{h_{n1}}) = 1$, então ou existe um número par de termos $x_i^{h_{ij}} = -1$ ou cada $x_i = -1$ está associado com uma potência $h_{ij} = 0$.

Se o primeiro caso ocorre, pela mudança de alfabeto como $1 \longmapsto -1$ e $0 \longmapsto 1$, temos que um número par de elementos 1 somados em $\mathbb{Z}_2$ é igual a zero. Ainda, se existirem termos $x_i^{h_{ij}}$ iguais a 1, então ou $x_i = 1$ e $h_{ij} = 1$ ou $x_i = -1$ e $h_{ij} = 0$. Em qualquer caso, pela mudança de alfabeto, cada entrada x_i da palavra do código será igual a zero. Logo,

$$\begin{pmatrix} x_1h_{11} & + & x_2h_{21} & + & \ldots & + & x_n0 \\ x_1h_{12} & + & x_2h_{22} & + & \ldots & + & x_n0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ x_1h_{1n-k} & + & x_2h_{2n-k} & + & \ldots & + & x_n1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

o que implica $x \cdot H^T = \overrightarrow{0}$ e, portanto, $x \in C$. O segundo caso, em que cada $x_i = -1$ está associado à uma potência $h_{ij} = 0$ implicará em $x_i \cdot h_{ij} = 1 \cdot 0 = 0$ e o resultado segue.

■

O Lema 3.2 garante que, no polinômio $\tilde{\text{E}}$, toda palavra do código corresponde a um máximo global (estado estável). O que queremos responder agora é: todos os máximos locais correspondem a uma palavra do código? O teorema 3.5 responde a esta pergunta. Antes disto, definimos:

Definição 3.16 *Dizemos que $v \in \mathbb{K}^n$, $\mathbb{K}$ corpo, é um* ***máximo local*** *para $\tilde{E}$ se existe $D(v,t) \in \mathbb{K}^n$, $t > 0$, tal que $max\{\tilde{E}(u)|u \in D(v,t)\} = \tilde{E}(v)$.*

Teorema 3.5 *[6, Teorema 3] Seja C um código de bloco linear, com G, H, E_C e $\tilde{E}$ como definidos acima. Então $\tilde{E}$ é um polinômio com as propriedades de E_C, isto é, x corresponde a um máximo local em $\tilde{E}$, se e somente se, $x \in C$.*

Demonstração: ($\Rightarrow$) Como H está na forma padrão, as últimas $n-k$ variáveis em $\tilde{E}$, $x_{k+1}, \ldots, x_n$, aparecem cada uma em apenas um termo, isto é, x_{k+1} aparece somente no 1° termo, x_{k+2} aparece somente no 2° termo e assim por diante (observe (3.26) no Lema 3.2). Assuma que exista um vetor v que corresponda a um máximo local, que não seja global, isto é, $\tilde{E}(v) = L < n-k$. Logo, existe $t \geq 1$ tal que em $D(v,t)$, $\tilde{E}(v)$ é máximo e, por (3.27) existe pelo menos um termo em $\tilde{E}$ que não seja igual a 1.

Sem perda de generalidade, podemos tomar este termo igual a $(x_1^{h_{11}} x_2^{h_{21}} \ldots x_k^{h_{k1}} x_{k+1})$. Se trocarmos a coordenada $k+1$ de v de sinal, a coordenada que aparece apenas neste termo, teremos $v' \in \mathbb{K}^n$ tal que $\tilde{E}(v') = L+1$ e $v' \in D(v,t)$, o que contradiz o fato de v ser um máximo local. Desta maneira, qualquer máximo local v deve ser tal que $\tilde{E}(v) = n-k$ e daí $v \in C$, pelo Lema 3.2.

($\Leftarrow$) Seja $x \in C$, pelo Lema 3.2, $\tilde{E}(x) = n-k$, ou seja, o máximo global de $\tilde{E}$ é $n-k$, assim cada palavra do código corresponde a um máximo global (e um local).

■

Exemplo 3.15 *Considere o $[n, n-1]$-código com um único dígito de verificação tal que*

$$G = [I_{n-1} : 1_{n-1}] \ \ e$$

$$H^T = 1_n$$

com 1_n o vetor de comprimento n com todas as entradas iguais a 1. Como os parâmetros de H^T são $n' = 1$ e $k' = n$, então

$$\tilde{E}(x) = \sum_{j=1}^{n'} w_j \prod_{i=1}^{k'} x_i^{h_{ij}} = w_1(x_1^{h_{11}} x_2^{h_{21}} \ldots x_n^{h_{n1}}) = x_1 x_2 \ldots x_n.$$

Assim $\tilde{E}(x) = 1$ se, e somente se, $x \in C$ pois, pelo Lema (3.2), $\tilde{E}(x) = n-k$ se, e somente se, $x \in C$ e como os parâmentros do código são n e $k = n-1$ então

$$\tilde{E}(x) = 1 = n - (n-1) \Rightarrow x \in C.$$

Ainda, $\tilde{E}(x) = -1$, para todo $x \notin C$ uma vez que $\tilde{E}$ assume apenas os valores 1 e -1, donde segue que o máximo local em $\tilde{E}$ tem uma bijeção com as palavras do código C, ou seja, para cada máximo local em $\tilde{E}$ existe uma palavra do código correspondente.

Exemplo 3.16 *Considere o código de repetição simples, isto é, um $[n, 1]$-código tal que*

$$G = [1, 1, \ldots, 1] \ \ e$$

$$H^T = [1_{n-1} : I_{n-1}] = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}_{n\times(n-1)}$$

Então

$$\tilde{E}(x) = \sum_{j=1}^{n} w_j \prod_{i=1}^{n-1} x_i^{g_{ij}} = w_1(x_1^{h_{11}} x_2^{h_{21}} \dots x_{n-1}^{h_{n-11}}) + \dots + w_n(x_1^{h_{1n}} x_2^{h_{2n}} \dots x_{n-1}^{h_{n-1n}})$$

$$\Rightarrow \tilde{E}(x) = x_1(x_2 + \dots + x_n), \quad \textit{para} \quad w = (1, 1, \dots, 1).$$

Observe que $\tilde{E}$ possui dois pontos nos quais atinge um máximo local, a saber, $x = (1, 1, \dots, 1)$ e $x = (-1, -1, \dots, -1)$.

Dado um código linear C, o algoritmo procurado no início desta seção para construir um polinômio E_C pode ser resumido como segue:

1) Construa a matriz geradora do código C na forma padrão.

2) Construa a matriz teste de paridade de C na forma padrão de acordo com (3.23).

3) Construa $\tilde{E}$, que é a representação polinomial de H^T, com $w = (1, 1, \dots, 1)$. Pelo Teorema 3.5, $E_C = \tilde{E}$.

Neste capítulo estabelecemos uma equivalência entre a decodificação de máxima verossimilhança de códigos corretores de erros e o processo de encontrar o máximo global de uma função de energia em uma rede neural. No próximo capítulo, construiremos um exemplo de decodificação utilizando redes neurais fazendo um paralelo entre a decodificação por síndrome da Teoria Clássica e um método utilizando redes.

Capítulo 4

Um exemplo de decodificação utilizando redes neurais

A utilização das redes neurais na decodificação de códigos corretores de erros possui dentre seus objetivos tornar o algoritmo de decodificação mais fácil de ser implementado e também mais rápido computacionalmente. Neste capítulo faremos um exemplo que ilustra tal utilização para a decodificação de um código linear que, em sua simulação, apresentou tais características [16], [21].

O objetivo deste capítulo, ao exibir tal exemplo, consiste ainda em explorar alguns dos conceitos envolvidos no processo de decodificação de códigos corretores de erros por meio de redes neurais artificiais.

Assim, abordaremos inicialmente alguns conceitos sobre as *redes perceptron* e o problema do ou-exclusivo. Após, faremos um exemplo utilizando o processo de decodificação por síndrome dos códigos corretores de erros e por fim construiremos a rede neural proposta e o exemplo que ilustra tal construção na decodificação de um código corretor de erros.

O decodificador (algoritmo de decodificação) de um código linear será simulado por uma rede neural que será adaptada para detectar e corrigir um código linear que possui no máximo um erro e cada camada de uma rede neural simulará um estágio do decodificador do código de bloco linear. Desta forma, a geração do vetor síndrome, a detecção de erros e os estágios de correção de erros do decodificador do código linear serão simulados pela rede neural proposta. As principais referências para as seções 4.1 e 4.2 são [17], [29] e [41].

4.1 Redes *Perceptron*

A forma mais simples de configuração de uma rede neural artificial é chamada de ***Perceptron.*** O *perceptron* foi idealizado por Rosenblatt em 1958 e é considerada simples por ser uma rede constituída de apenas uma camada neural, tendo um único neurônio artificial nesta camada, com pesos sinápticos e limiar de ativação.

O *perceptron* é construído tendo como referência um modelo de neurônio não linear, o modelo de McCulloch e Pitts (ver página 19), ele é uma extensão natural deste neurônio [2]. Desta forma, seu princípio de funcionamento assemelha-se ao mesmo já descrito para os neurônios artificiais na Seção 2.1.2. Cada uma das entradas x_i representa uma informação sobre o comportamento do processo a ser mapeado e inicialmente será ponderada pelo peso sináptico w_i que quantifica a importância de cada uma, de acordo com os

objetivos funcionais atribuídos ao neurônio, cujo propósito será mapear o comportamento entrada/saída do referido processo.

O valor resultante da composição de todas as entradas já devidamente ponderadas pelos seus respectivos pesos, adicionado ainda do limiar de ativação θ, é repassado como argumento da função de ativação, cujo resultado será a saída y produzida pelo *perceptron* [41, p. 59].

O processo interno de uma rede *perceptron* pode ser descrito como:

$$\begin{cases} u = \sum_{i=1}^{n} w_i.x_i - \theta, \\ y = g(u) \end{cases}$$

e alguns aspectos da rede *perceptron* podem ser listados:

1. Devido às características estruturais, as funções de ativação frequentemente utilizadas na rede *perceptron* são a função degrau e a bipolar, existindo assim apenas duas possibilidades de valores a serem produzidos pela saída, usando o alfabeto binário $\{0, 1\}$ ou a forma bipolar $\{1, -1\}$.

2. O *perceptron* é a forma mais simples de configuração de uma rede neural utilizada para classificação de padrões ditos linearmente separáveis, isto é, padrões que se encontram em lados opostos de uma reta (ou um hiperplano para o caso $n-$dimensional).

   Como sua saída pode assumir somente dois valores possíveis, associa-se a cada um de tais valores uma das classes que o *perceptron* está identificando.

   Assim, por exemplo, dado um problema envolvendo a classificação dos sinais de entrada em duas classes, classe A e classe B, seria possível atribuir o valor -1 para representar amostras pertencentes a classe A e 1 para amostras pertencentes a classe B. Logo, como na Figura 4.1, o *perceptron* é capaz de dividir duas classes linearmente separáveis de forma que quando a saída for -1 os padrões (classe A) estarão localizados acima da fronteira (reta) de separação, ao passo que quando a saída for igual a 1 os padrões (classe B) estarão abaixo desta fronteira.

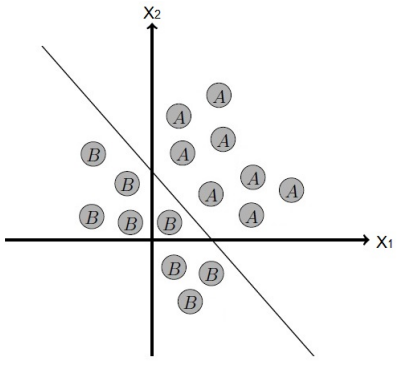


Figura 4.1: Fronteira de separação (reta) das classes A e B considerando um neurônio com duas entradas.

3. O ajuste dos pesos e limiar do *perceptron* é feito utilizando o processo de treinamento supervisionado, visto ao final da Seção (2.1.4). A regra de aprendizado realizada é conhecida como regra de aprendizado de *Hebb*. Assim, se a saída produzida pelo

perceptron coincide com a saída desejada, os pesos sinápticos e os limiares da rede são incrementados proporcionalmente aos valores de seus sinais de entrada. Porém, se a saída produzida pela rede é diferente do valor desejado, os pesos sinápticos e o limiar serão decrementados. Tal processo é repetido sequencialmente para todas as amostras de treinamento até que a saída produzida pelo *perceptron* seja similar à saída desejada de cada amostra.

A condição para que o *perceptron* de camada simples possa ser utilizado como um classificador de padrões consiste em que as classes do problema a ser mapeado sejam linearmente separáveis. Tal princípio condicional foi enunciado como *Teorema de Convergência do Perceptron*.

Todavia, padrões de entrada que sejam não linearmente separáveis ocorrem com frequência como por exemplo, no problema do ou-exclusivo (porta lógica X_{OR}), do qual falaremos com mais detalhes na próxima seção. Para resolver tal problema, pode-se inserir a rede *perceptron* multicamadas.

As redes ***perceptron* multicamadas** são utilizadas na solução de diversos problemas como os de classificação e reconhecimento de padrões, problemas relacionados à aproximação de funções e os direcionados aos sistemas dinâmicos. Elas pertencem à arquitetura *feeedforward* de camadas múltiplas e representam uma generalização do *perceptron* de camada única.

O treinamento das redes *perceptron* multicamadas é feito de forma supervisionada por meio do algoritmo conhecido como *algoritmo de retropropagação de erro* (*error back-propagation*). Este algoritmo é baseado na regra de aprendizagem por correção de erro [17, pp. 76, 183], que em resumo, consiste de dois passos: um *passo para frente* (*propagação*) que envia para frente os sinais da rede, uma amostra do conjunto de treinamento propagando os sinais de entrada, camada a camada, até a produção das respectivas saídas, mantendo fixos os pesos sinápticos, e um *passo para trás* (*retropropagação*) ajustando os pesos sinápticos de forma a mover a resposta real da rede para mais perto da resposta desejada.

No primeiro passo, as respostas produzidas pela rede são comparadas com as respostas desejadas e os respectivos erros entre estas respostas são calculados e utilizados para ajustar os pesos e limiares de todos os neurônios.

Diferente da rede *perceptron* de camada simples, a rede *perceptron* de camadas múltiplas pode ter diversos neurônios na camada de saída. Os sinais de entrada da rede partindo da camada de entrada se propagam para frente, camada por camada, e as camadas intermediárias extraem a maioria das informações codificando-as por meio de pesos sinápticos e limiares formando uma representação própria do ambiente em que está inserido o sistema a ser tratado. Os neurônios da camada de saída recebem os estímulos advindos dos neurônios da última camada intermediária, produzindo um padrão de resposta que será a saída disponibilizada pela rede.

4.2 O problema do ou-exclusivo

Por não possuir neurônios ocultos, o *perceptron* de camada única não é capaz de classificar padrões de entrada que não sejam não linearmente separáveis. Um exemplo de tal problema é *o problema do ou-exclusivo*, ou problema do X_{OR}, que pode ser visto como um problema de classificação de pontos, como descreveremos a seguir.

Considere os quatro vértices do quadrado unitário, Figura 4.2, que correspondem aos padrões de entrada $(0,0),(0,1),(1,1)$ e $(1,0)$. Note que o primeiro e o terceiro padrão, isto é, $(0,0)$ e $(1,1)$ pertencem a classe 0 (considerando a função booleana ou-exclusivo), pois

$$0 \oplus 0 = 0 \;\text{ e }\; 1 \oplus 1 = 0$$

Assim, embora produzam a mesma saída, igual a zero, estes padrões de entrada estão em vértices opostos do quadrado unitário. O mesmo acontece com $(0,1)$ e $(1,0)$ que pertencem a classe 1 :

$$0 \oplus 1 = 1 \;\text{ e }\; 1 \oplus 0 = 1$$

Deste modo, não podemos construir uma linha reta como uma fronteira de decisão de forma a colocar os padrões em uma mesma classe, conforme mostra a Figura 4.2.

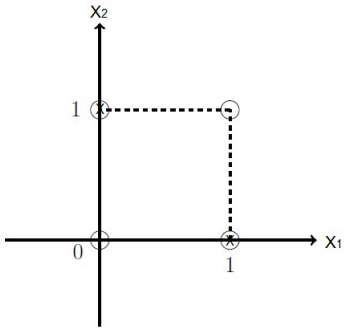


Figura 4.2: Ilustração do problema do ou-exclusivo.

Todavia, este problema pode ser resolvido utilizando uma camada oculta com dois neurônios,conforme ilustrado na Figura 4.3.

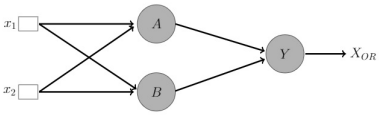


Figura 4.3: Rede *perceptron* multicamadas aplicada no problema ou-exclusivo.

Com base neste neurônio, o problema do ou-exclusivo pode ser solucionado conforme é visto na Figura 4.4.

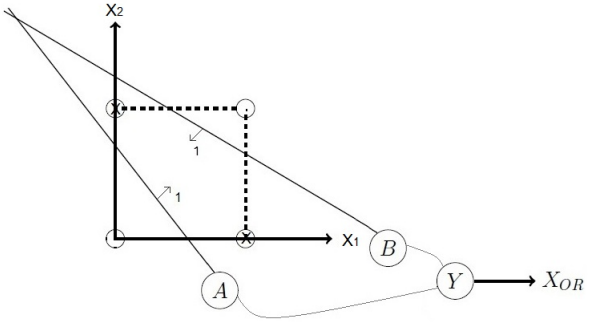


Figura 4.4: Separabilidade no problema ou-exclusivo.

Deste modo, assumindo uma função lógica em cada neurônio tem-se que o neurônio A terá uma saída igual a 1 apenas para os padrões que estejam acima de sua reta, o neurônio B terá a saída igual a 1 para todos os padrões que estejam abaixo da sua reta e o neurônio Y, efetuando a operação lógica E, produzirá a saída 1 quando, em ambos casos as saídas, forem iguais a 1, e zero, caso contrário. Vejamos isto com mais detalhes.

4.2.1 Operadores lógicos e redes neurais

Em 1943, McCulloch e Pitts mostraram como os neurônios poderiam implementar operações lógicas. Faremos aqui uma breve descrição desta relação para as operações "E", "OU" e *"NÃO"* e discutiremos ao final uma solução para o problema do ou-exclusivo.

Suponha A e B proposições e considere a tabela verdade dos operadores lógicos "E" ($\wedge$), "OU" ($\vee$) e do operador negação *"NÃO"*($\neg$) dadas em 4.1, 4.2 e 4.3 respectivamente.

A	B	A $\wedge$ B
0	0	0
0	1	0
1	0	0
1	1	1

Tabela 4.1: Tabela-verdade "E".

A	B	A $\vee$ B
0	0	0
0	1	1
1	0	1
1	1	1

Tabela 4.2: Tabela-verdade "OU".

A	$\neg$ A
1	0
0	1

Tabela 4.3: Tabela-verdade do operador de negação "NÃO".

• **Operador "E"**

Consideremos agora as proposições A e B como as entradas de um neurônio em uma rede neural. Observando a tabela verdade (4.1) do operador lógico "E" tem-se que um neurônio deve produzir o valor 1 apenas quando ambas entradas produzem o valor 1 (podemos dizer que ambas entradas estão ativas). Contudo, para que isto ocorra é necessário que a soma das entradas seja maior que o valor do limiar de ativação[1], mas cada entrada sozinha precisa ser menor que este valor.

Desta forma, utilizando um limiar de valor igual a 1, por simplicidade, podemos escolher os pesos, por exemplo, com valores iguais a 0.6. Com isto, a ativação individual da entrada A ou B não excederá o limite, enquanto a soma das duas será 1.2, o que excede o limite e faz o neurônio disparar. De fato,

Seja $\theta = 1$ o valor limite e considere as entradas do neurônio A e B como na tabela verdade do operador "E",

1. Se $A = 0$ e $B = 0$ então $A \cdot w_1 + B \cdot w_2 = 0 \cdot 0.6 + 0 \cdot 0.6 = 0$ e como $0 \leq \theta = 1$, isto

[1]Descreveremos durante o texto o limiar de ativação também como valor limite ou apenas como limiar.

é, o valor da soma é menor do que o valor do limite θ, tem-se que o valor de saída é igual a zero.

2. Se $A = 0$ e $B = 1$ então $A \cdot w_1 + B \cdot w_2 = 0 \cdot 0.6 + 1 \cdot 0.6 = 0.6 \Longrightarrow 0.6 \leq \theta = 1$ e o valor de saída é igual a zero.

3. Se $A = 1$ e $B = 0$ então $A \cdot w_1 + B \cdot w_2 = 1 \cdot 0.6 + 0 \cdot 0.6 = 0.6 \Longrightarrow 0.6 \leq \theta = 1$ e o valor de saída é igual a zero.

4. Se $A = 1$ e $B = 1$ então $A \cdot w_1 + B \cdot w_2 = 1 \cdot 0.6 + 1 \cdot 0.6 = 1.2 \Longrightarrow 1.2 \geq \theta = 1$ e o valor de saída é igual a um.

O que de fato condiz com a tabela verdade do operador lógico "E". Observe que se tivéssemos tomado, por exemplo, o peso da sinapse que conecta o neurônio A com o neurônio de saída com valor igual a 1, quando $A = 1$ e $B = 0$ teríamos

$$A \cdot w_1 + B \cdot w_2 = 1 \cdot 1 + 0 \cdot 0.6 = 1 \geq \theta = 1$$

o que implica que o neurônio seria disparado, contradizendo a tabela do operador "E", ou seja, com um valor de peso maior tal neurônio não seria capaz de implementar o operador lógico "E".

Um diagrama representando a implementação deste operador pode ser visto na Figura 4.5.

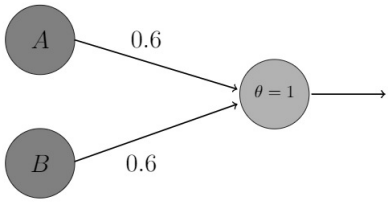


Figura 4.5: Representação da implementação do operador lógico "E".

• **Operador "OU"**

Observando a tabela verdade (4.2) do operador lógico "OU" queremos que o valor de saída seja 1 quando uma, ou ambas, entradas A e B estiverem ativas, e igual a 0 quando ambas entradas estiverem inativas (desligadas), isto é, quando o valor dessas entradas for igual a zero.

Assim, se o valor de cada sinapse for maior que o valor limite, o neurônio irá disparar sempre que houver alguma atividade em uma entrada ou em todas.

Por exemplo, tomando $\theta = 1$ e cada peso da sinapse igual a 1.1 temos:

1. Se $A = 0$ e $B = 0$ então $A \cdot w_1 + B \cdot w_2 = 0 \cdot 1.1 + 0 \cdot 1.1 = 0 \leq \theta = 1$, o que implica no valor de saída ser igual a zero.

2. Se $A = 0$ e $B = 1$ então $A \cdot w_1 + B \cdot w_2 = 0 \cdot 1.1 + 1 \cdot 1.1 = 1.1 \geq \theta = 1$ e o valor de saída é igual a um. O mesmo se aplica ao caso $A = 1$ e $B = 0$ e ao caso $A = 1$ e $B = 1$.

Um diagrama representando a implementação do operador "OU" pode ser visto na Figura 4.6.

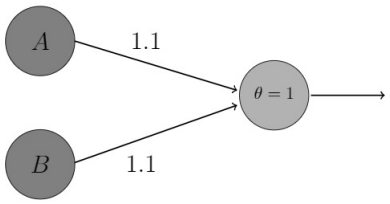


Figura 4.6: Representação da implementação do operador lógico "OU".

• **Operador "NÃO"**

Observando agora a tabela verdade do operador *"NÃO"*, nota-se que neste caso precisamos mudar o valor de entrada 1 para o valor de saída 0 e, por outro lado, o valor de entrada zero precisa ser convertido no valor de saída igual a um. Assim, temos dois casos a considerar:

1. **Mudança** $1 \longrightarrow 0$:

   Neste primeiro caso, para alterar a entrada 1 no valor de saída 0 basta fazer com que o peso da sinapse não exceda o limite.

2. **Mudança** $0 \longrightarrow 1$:

   Neste segundo caso, para alterar a entrada 0 no valor de saída 1 precisamos pensar no neurônio cujo estado natural é ativo (ou seja, valor igual a 1). Desta forma é preciso definir um limite com valor abaixo de zero e deste modo, ainda que a entrada seja zero a soma sempre excederá o limite fazendo com que o valor de saída seja igual a 1.

Assim, definimos o peso da sinapse como sendo um número negativo e o valor limite como sendo algum valor entre o peso da sinapse e zero, por exemplo:

Seja $\theta = -0.5$ e $w_{AS} = -1$ (S denotando o neurônio de saída). Se $A = 1$, então $w_{AS} \cdot A = -1 \cdot 1 = -1 < -0.5$ e o neurônio não será disparado produzindo valor de saída igual a zero.

Se $A = 0$, então $w_{AS} \cdot A = -1 \cdot 0 = 0 \geq -0.5$ e o neurônio será disparado produzindo valor de saída igual a um.

Um diagrama representando a implementação do operador "NÃO" pode ser visto na Figura 4.7.

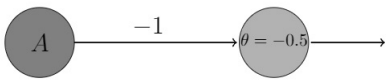


Figura 4.7: Representação da implementação do operador lógico "NÃO".

• **Operador ou-exclusivo** (X_{OR})

Por bastante tempo o operador ou-exclusivo foi um empecilho para as pesquisas envolvendo redes neurais uma vez que não era possível criar uma porta lógica (X_{OR}) com um único neurônio, ou mesmo uma única camada de neurônios.

Observando a Tabela verdade, Tabela (4.4), do operador ou-exclusivo temos que a proposição $A\ X_{OR}\ B$ é verdadeira quando a proposição A é verdadeira e a proposição B é falsa ou quando a proposição B é verdadeira e a proposição A é falsa. Porém, a proposição $A\ X_{OR}\ B$ é falsa quando ambas proposições, A e B, são ao mesmo tempo verdadeiras ou falsas.

x_1	x_2	$x_1\ X_{OR}\ x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 4.4: Tabela-verdade ou-exclusivo.

Deste modo, para configurar um neurônio para realizar a operação (X_{OR}) precisamos fazer com que vários neurônios trabalhem juntos (teremos assim duas camadas de neurônios). Para isto utilizaremos as portas lógicas OU, E e *"NÃO"* operando juntas.

Inicialmente, dividindo a operação (X_{OR}) em funções lógicas mais simples:

$$A\ X_{OR}\ B = (A \vee B) \wedge \neg(A \wedge B), \tag{4.1}$$

ou seja, $A\ X_{OR}\ B$ é o mesmo que A ou B e não A e B. Logo, podemos criar um neurônio para cada uma destas operações e juntá-las como na Figura 4.8.

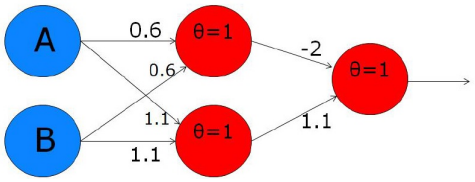


Figura 4.8: Representação da implementação do operador lógico "XOR".

Na rede neural representada na Figura 4.8, observe que o neurônio em vermelho na parte superior da segunda camada (denotado por $S1$) possuí duas entradas A e B com pesos sinápticos 0.6 e limiar $\theta = 1$, o que é exatamente a mesma porta lógica E mostrada na Figura 4.5. Deste modo, temos representada a função lógica $(A \wedge B)$ da equação (4.1).

O neurônio vermelho inferior da segunda camada (denotado por $S2$), cujos pesos sinápticos com relação as entradas A e B tem valor igual a 1.1 e seu limiar é dado por $\theta = 1$, é também o mesmo neurônio apresentado na Figura 4.6 que representa a implementação da porta lógica OU. Este neurônio realiza a função lógica $(A \vee B)$ na equação (4.1).

O neurônio de saída está executando outra operação E, a operação feita entre $(A \vee B)$ e $\neg(A \wedge B)$. Desta forma, este neurônio será disparado sempre que uma das entradas A ou B estiver ativa. Porém, pela inibição do neurônio superior, o neurônio de saída estará inativo (não será disparado) nos casos em que A e B estiverem ambos ativos. Analisando as entradas temos:

1. Se $A = 0$ e $B = 0$, então

$$A \cdot w_{AS1} + B \cdot w_{BS1} = 0 \cdot 0.6 + 0 \cdot 0.6 = 0 < \theta \Longrightarrow \text{ saída igual a zero.}$$

$$A \cdot w_{AS2} + B \cdot w_{BS2} = 0 \cdot 1.1 + 0 \cdot 1.1 = 0 < \theta \Longrightarrow \text{ saída igual a zero.}$$

Logo, a saída da rede calculada por $0 \cdot (-2) + 0 \cdot 1.1 = 0 < \theta$ será igual a zero.

2. Se $A = 0$ e $B = 1$, então

$$A \cdot w_{AS1} + B \cdot w_{BS1} = 0 \cdot 0.6 + 1 \cdot 0.6 = 0 < \theta \Longrightarrow \text{ saída igual a zero.}$$

$$A \cdot w_{AS2} + B \cdot w_{BS2} = 0 \cdot 1.1 + 1 \cdot 1.1 = 0 \geq \theta \Longrightarrow \text{ saída igual a um.}$$

Donde segue $0 \cdot (-2) + 1 \cdot 1.1 = 0 \geq \theta$ e a saída da rede é igual a um. O mesmo processo ocorre para $A = 1$ e $B = 0$.

3. Se $A = 1$ e $B = 1$ então

$$A \cdot w_{AS1} + B \cdot w_{BS1} = 1 \cdot 0.6 + 1 \cdot 0.6 = 1.2 > \theta \Longrightarrow \text{ saída igual a um.}$$

$$A \cdot w_{AS2} + B \cdot w_{BS2} = 1 \cdot 1.1 + 1 \cdot 1.1 = 2.2 > \theta \Longrightarrow \text{ saída igual a um.}$$

Como $1 \cdot (-2) + 1 \cdot 1.1 = -0.9 < \theta$, a saída da rede é igual a zero. Portanto, o neurônio representado na Figura 4.8 implementa a porta lógica X_{OR}.

4.3 Decodificador de um código de bloco linear

Chama-se *decodificação* ao procedimento de detecção e correção de erros em um determinado código. Descrevemos nesta seção um algoritmo de decodificação para um código linear corretor de um erro. Tal método é um aperfeiçoamento do método criado pelo matemático David Slepian do Laboratório Bell, na década de 60.

O objetivo em descrever este algoritmo é simular posteriormente os mesmos estágios neste processo de decodificação com a rede neural proposta. Por isto, para fins comparativos, o processo do decodificação será dividido em três fases: fase da síndrome, fase de detecção de erro e fase de correção de erro.

Inicialmente, define-se o **vetor erro** "**e**" como sendo a diferença entre o vetor recebido "r" e o vetor transmitido "c", isto é,

$$\mathbf{e = r - c}.$$

Exemplo 4.1 *Seja $\mathscr{C}$ um código definido sobre $\mathbb{F}_2$ e suponha que se tenha transmitido a palavra* (0 1 0 1 0 1) *e a palavra recebida tenha sido* (0 1 0 1 1 1). *Então o vetor erro é dado por*

$$\boldsymbol{e} = (0\ 1\ 0\ 1\ 1\ 1\ ) - (0\ 1\ 0\ 1\ 0\ 1\ ) = (0\ 0\ 0\ 0\ 1\ 0\ ).$$

Note que o peso do vetor erro determina quantos erros foram cometidos desde a transmissão até a recepção da palavra do código. Assim, no Exemplo 4.1

$$w(0\ 0\ 0\ 0\ 1\ 0\ ) = w(\mathbf{e}) := |\{i; \mathbf{e}_i \neq 0\}| = 1.$$

Se H é a matriz teste de paridade de um código C, então $Hc^t = 0$, para todo $c \in C$. Com isso,

$$H\mathbf{e}^t = H(r^t - c^t) = Hr^t - Hc^t = Hr^t.$$

Logo o vetor erro $\mathbf{e}$ e a palavra recebida r têm a mesma síndrome.

Para melhor entendimento do lema seguinte, denotemos por h^i a i-ésima coluna de H. Se $\mathbf{e} = (\alpha_1, \ldots, \alpha_n)$, então

$$\sum_{i=1}^{n} \alpha_i h^i = H\mathbf{e}^t = Hr^t.$$

Lema 4.1 *[18, Lema 4] Seja $C \subset \mathbb{K}^n$ um código linear com capacidade de correção κ. Se $r \in \mathbb{K}^n$ e $c \in C$ são tais que $d(c, r) \leq \kappa$, então existe um único vetor $\boldsymbol{e}$ com $\omega(\boldsymbol{e}) \leq \kappa$, cuja síndrome é igual à síndrome de r e $c = r - \boldsymbol{e}$.*

De fato, se $\mathbf{e} = r - c$, então

$$\omega(\mathbf{e}) = \omega(r - c) = d(r, c) \leq \kappa.$$

Logo $\omega(\mathbf{e}) \leq \kappa$. Para provar a unicidade, sejam $\mathbf{e} = (\alpha_1, \ldots, \alpha_n)$ e $\mathbf{e'} = (\alpha'_1 \ldots \alpha'_n)$ tais que $\omega(\mathbf{e}) \leq \kappa$ e $\omega(\mathbf{e'}) \leq \kappa$, ambos com a mesma síndrome de r.

Como H é a matriz teste de paridade de C, temos

$$H\mathbf{e}^t = H\mathbf{e'}^t \implies \sum_{i=1}^{n} \alpha_i h^i = \sum_{i=1}^{n} \alpha'_i h^i \implies \sum_{i=1}^{n} (\alpha_i - \alpha'_i) h^i = 0,$$

nos dando uma relação de dependência linear entre $1 \leq 2\kappa \leq d - 1$ colunas de H. Como quaisquer $d - 1$ colunas de H são linearmente independentes, temos

$$\alpha_i - \alpha'_i = 0 \Rightarrow \alpha_i = \alpha'_i, \text{para todo } i.$$

Logo $\mathbf{e} = \mathbf{e'}$.

■

Como determinar tal vetor $\mathbf{e}$ a partir de $H\mathbf{e}^t$, quando $\omega(\mathbf{e}) \leq 1$?

Suponha que o código C tenha distância mínima $d \geq 3$ e que o vetor erro $\mathbf{e}$, introduzido entre a palavra transmitida c e a palavra recebida r, tenha peso $\omega(\mathbf{e}) \leq 1$. Se $H\mathbf{e}^t = H\mathbf{r}^t = 0$, então $r \in C$ e tome $c = r$. Se $H\mathbf{e}^t = H\mathbf{r}^t \neq 0$, então $\omega(\mathbf{e}) = 1$, o que nos diz que $\mathbf{e}$ tem apenas uma coordenada não nula. Considerando $\mathbf{e} = (0, \ldots, \alpha, \ldots, 0)$, $\alpha \neq 0$ na i-ésima coordenada, temos

$$H\mathbf{e}^t = \alpha h^i,$$

com h^i a i-ésima coluna de H. Assim, $\mathbf{e}$ é o vetor com todas entradas nulas exceto a i-ésima coordenada, que é α. Observe que $\mathbf{e}$ está bem determinado, com $i \neq j$, pois não corremos o risco de $h^i = h^j$ em H, já que supomos $d \geq 3$.

Com isto, podemos estabelecer o algoritmo de decodificação em códigos corretores de um erro.

Sejam H a matriz teste de paridade do código $C \subset \mathbb{K}^n$ e $\mathbf{r}$ um vetor recebido. Suponha $d \geq 3$.

A) Fase da síndrome

Sendo r a palavra transmitida a síndrome é calculada:

1. Calcule $H\mathbf{r}^t$.
2. Se $H\mathbf{r}^t = 0$, aceite $\mathbf{r}$ como sendo a palavra enviada.

B) Fase de detecção de erro

De acordo com a síndrome encontrada na Fase A, o vetor erro pode ser encontrado:

3. Se $H\mathbf{r}^t = \mathbf{s}^t \neq 0$, compare $\mathbf{s}^t$ com as colunas de H.
4. Se existirem i e α tais que $\mathbf{s}^t = \alpha h^i, \alpha \in \mathbb{K}$, então $\mathbf{e}$ é a n-upla com α na posição i e zeros nas outras coordenadas.

C) Fase de correção do erro

5. Corrija $\mathbf{r}$ pondo $\mathbf{c} = \mathbf{r} - \mathbf{e}$.
6. Se o contrário de (4) acontecer, então mais de um erro foi cometido.

Exemplo 4.2 *Considere o* $(6, 3)$*-código linear cuja matriz teste de paridade é dada por:*

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

e seja $r = (1\ 0\ 0\ 1\ 1\ 1\ )$ *a palavra recebida. Vamos encontrar a palavra transmitida* c*.*

A) Fase da síndrome

1. Calcule $H\mathbf{r}^t = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$.
2. Como $\mathbf{s}^t = H\mathbf{r}^t \neq 0$ passemos para a segunda fase.

B) Fase de detecção de erro

3. Comparando $\mathbf{s}^t$ com as colunas de H temos $\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = 1 \cdot h^6$
4. Portanto $\mathbf{e}$ é a n-upla com $\alpha = 1$ na posição $i = 6$ e zeros nas outras coordenadas, isto é, $e = (0\ 0\ 0\ 0\ 0\ 1\ )$.

C) Fase de correção do erro

5. Fazendo $\mathbf{c} = \mathbf{r} - \mathbf{e} = (1\ 0\ 0\ 1\ 1\ 1) - (0\ 0\ 0\ 0\ 0\ 1\ ) = (1\ 0\ 0\ 1\ 1\ 0)$ obtemos a palavra transmitida.

   Por outro lado, a palavra do código transmitida **c** pode ser encontrada, dependendo do erro obtido na Fase B, por meio da operação **ou-exclusivo**:

r_i	$\oplus$	e_i	$=$	c_i
1	$\oplus$	0	$=$	1
0	$\oplus$	0	$=$	0
0	$\oplus$	0	$=$	0
1	$\oplus$	0	$=$	1
1	$\oplus$	0	$=$	1
1	$\oplus$	1	$=$	0

4.4 Construção da rede neural

Nesta seção, e na Seção 4.5, todas as três fases no processo de decodificação do código linear serão construídas pelo uso da rede neural.

De forma simplificada, a construção da rede neural é feita do seguinte modo: existem N entradas para a rede que consistem em memórias associativas[2] compostas por duas camadas de neurônios (camadas 1 e 2, Figura 4.9). O número N representa o comprimento de uma palavra do código. Na camada oculta (camada 2), existem M neurônios em que M representa o número de linhas na matriz teste de paridade H dada. Cada entrada é conectada com cada neurônio da camada 2. A camada 3 tem N neurônios que determinam a posição do erro da palavra recebida. Cada neurônio da camada 2 é conectado a cada neurônio da camada 3.

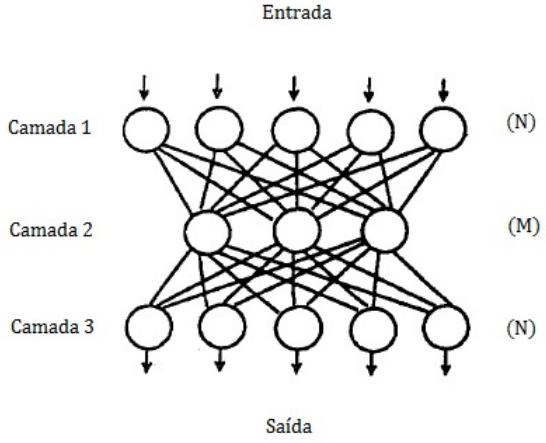


Figura 4.9: Rede neural [21].

Para o nosso exemplo, construiremos uma rede neural com cinco camadas. A primeira e a segunda camadas estimarão o vetor da síndrome, a segunda e a terceira camadas detectarão o erro e a terceira, quarta e quinta camadas corrigirão o erro, Figura 4.10.

Na Figura 4.10 note que a rede neural em linhas em negrito tem o objetivo de atrasar a entrada. Tal conceito será ilustrado no exemplo da Seção 4.6.

[2]Veja definição na página 32.

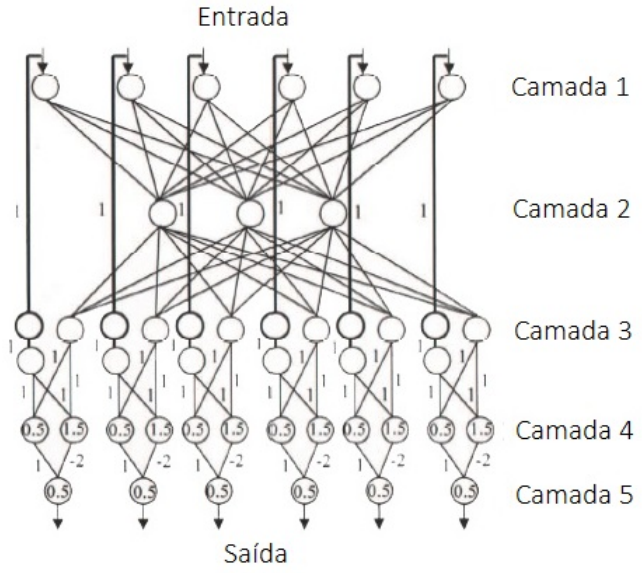


Figura 4.10: Rede neural [16].

4.5 Processo de decodificação por meio de redes neurais

Descrevemos aqui as três fases no processo de decodificação utilizando as redes neurais. Consideraremos um código C sobre $\mathbb{F}_2$, H a matriz teste de paridade dada e h_{ij} o elemento na i-ésima linha e j-ésima coluna de H.

Antes de especificar cada fase, definimos alguns parâmetros utilizados:

- Os neurônios da camada de entrada, camada 1, serão denotados por r_i^1, com $1 \leq i \leq n$. Cada r_i representa uma coordenada da palavra recebida $r = (r_1, r_2, \cdots, r_n)$.
- A saída do neurônio t na camada 2 será denotada por v_j^2, com $1 \leq j \leq k$.
- A saída do neurônio i na camada 3 será denotada por e_i^3, para $1 \leq i \leq n$, em que cada e_i é uma coordenada do vetor erro $e = (e_1, e_2, \cdots, e_n)$.

1. **Fase de geração da síndrome (camadas 1 e 2):**

   Esta fase é semelhante a primeira fase no decodificador de códigos lineares que é responsável por gerar o vetor síndrome **s**. Os elementos da matriz teste de paridade H são utilizados como os pesos sinápticos nas conexões dos vértices de entrada da primeira camada com os nós da segunda camada. Assim, denotamos o peso w_{ij}^{12} da conexão do i-ésimo neurônio da camada de entrada com o j-ésimo neurônio da camada 2, ou seja,

   $$w_{ij}^{12} = h_{ji}, \tag{4.2}$$

   com $1 \leq i \leq n$ e $1 \leq j \leq k$, em que n é o comprimento da palavra e k a dimensão do código. O sobrescrito ^{12} refere-se aos pesos sinápticos entre as camadas 1 e 2.

   Seja

   $$s_j^2 = \sum_{i=1}^{n} w_{ij}^{12} r_i^1, \tag{4.3}$$

para $1 \leq j \leq k$. A saída da camada 2 é dada por

$$v_j^2 = g_2(s_j^2) = \begin{cases} 1, & \text{se } s_j^2 \equiv 1(mod 2), \\ -1, & \text{caso contrário.} \end{cases} \tag{4.4}$$

Aqui g_2 é a função de ativação para o neurônio da camada 2, definida como a *função sinal* módulo 2 na soma ponderada das entradas r_i^1, Figura 4.11.

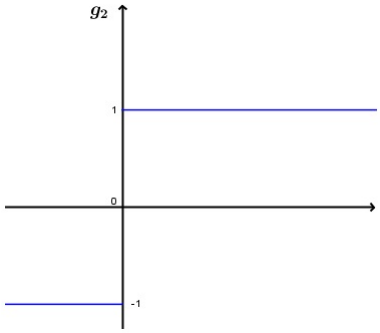


Figura 4.11: Função de ativação g_2 utilizada na camada 2 [16].

2. **Fase de detecção de erro (camadas 2 e 3):**

   Esta fase é semelhante a segunda fase do decodificador de códigos lineares que é responsável por detectar a posição do erro no código recebido.

   Para detectar o erro, a palavra recebida passa pela 2^a camada e permitimos que a rede avance até cair em uma situação estável. Neste caso, se um neurônio da camada 3 produzir valor igual a 1 enquanto os demais produzirem um valor igual a zero, então obtemos a posição do *bit* (coordenada) que foi transmitido de forma incerta.

   Se não houver erro na palavra recebida, todos os neurônios da camada 3 produzirão valores iguais a zero.

   O termo w_{ij}^{23} denota o peso da conexão do $i-$ésimo neurônio da camada 2 com o $j-$ésimo neurônio da camada 3. Os valores w_{ij}^{23} são também atribuídos pelos elementos da matriz teste de paridade H, mas na forma bipolar[3] (o dígito 0 é substituído por -1), isto é,

$$w_{ij}^{23} = \begin{cases} 1, & \text{se } h_{ij} = 1, \\ -1, & \text{se } h_{ij} = 0. \end{cases} \tag{4.5}$$

   Os valores de saída dos neurônios da camada 3 são determinados pela função degrau g_3, função de ativação da camada 3, mostrada na Figura 4.12. Assim, seja

$$s_i^3 = \sum_{j=1}^{k} w_{ji}^{23} v_j^2, \tag{4.6}$$

   para $1 \leq i \leq n$ e $\theta = k - \dfrac{1}{2}$. A saída da camada 3 é dada por

[3]É importante notar que a forma bipolar é diferente da mudança de alfabeto ($0 \longmapsto 1, \;\; 1 \longmapsto -1$) feita no Capítulo 3, página 44.

$$e_i^3 = g_3(s_i^3) = \begin{cases} 1, & \text{se } s_i^3 \geq \theta \\ 0, & \text{caso contrário.} \end{cases} \tag{4.7}$$

Em outras palavras, desde que o $i-$ésimo neurônio da camada 3 aceitar como entrada os valores s_i^3, com $s_i^3 = k$ e $s_i^3 > \theta$, a saída e_i^3 será igual a 1.

Para cada neurônio $j \neq i$ da camada 3, vale $s_j^3 < k - 1$ e $s_j^3 < \theta$. Portanto a saída e_i^3 pode ser igual a zero [11].

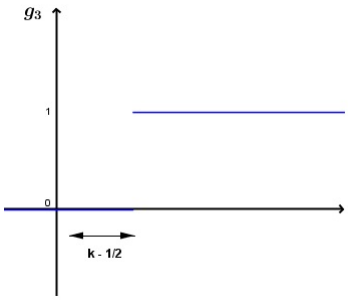


Figura 4.12: Função de ativação g_3 utilizada na camada 3 [16].

3. **Fase de correção de erros (camadas 3 e 4):**

   Nesta terceira fase utilizamos uma rede neural ou-exclusivo (X_{OR}) para corrigir a palavra do código. Existem muitos métodos para construir a rede (X_{OR}) [21]. Aqui adotamos o método utilizado em [16] e [21] mostrado na Figura 4.13.

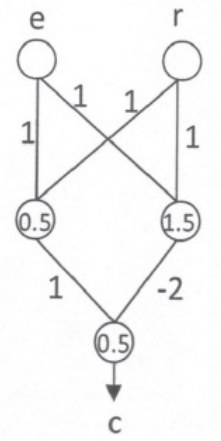


Figura 4.13: Rede Neural para XOR [16].

Para encontrarmos a palavra c do código, a saída da camada 3 é (X_{OR}) com a palavra recebida r. Assim, sendo $g_{3'}, g_4$, e g_5 as funções de ativação, Figuras 4.14, 4.15 e 4.16, respectivamente, temos

$$c_i = g_5(g_{3'}(e_i + r_i) - 2 \cdot g_4(e_i + r_i)) \tag{4.8}$$

em que

$$g_{3'} = \begin{cases} 1, & \text{se } (e + r) \geq 0.5, \\ 0, & \text{caso contrário.} \end{cases} \tag{4.9}$$

$$g_4 = \begin{cases} 1, & \text{se } (e+r) \geq 1.5, \\ 0, & \text{caso contrário.} \end{cases} \tag{4.10}$$

$$g_5 = \begin{cases} 1, & \text{se } (g_3(e+r) - 2 \cdot g_4(e+r)) \geq 0.5, \\ 0, & \text{caso contrário.} \end{cases} \tag{4.11}$$

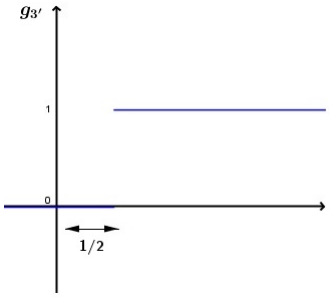


Figura 4.14: Função de ativação $g_{3'}$ utilizada na rede neural X_{OR} [16].

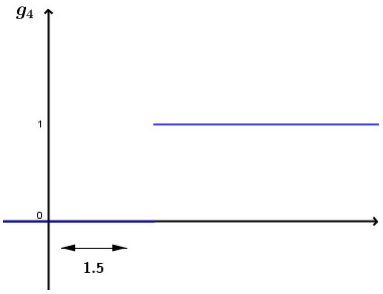


Figura 4.15: Função de ativação g_4 utilizada na rede neural X_{OR} [16].

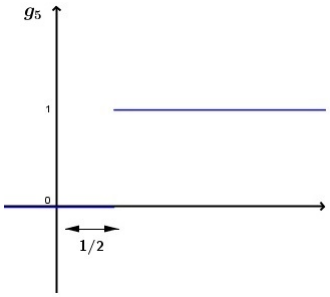


Figura 4.16: Função de ativação g_5 utilizada na rede neural X_{OR} [16].

Com a intenção de tornar mais claras as construções das funções de ativação $g_{3'}, g_4$ e g_5, feitas acima, observamos o seguinte.

Considere a rede neural de 5 camadas construída, Figura 4.10, e a rede X_{OR} mostrada na Figura 4.13. Seja $g_{3'}$ a função de ativação descrita em (4.9). Tomando v_j^3, com $1 \leq j \leq 2$, as entradas da camada 3 (o que é referente a primeira camada observando apenas a rede neural X_{OR}), podemos descrever o potencial de ativação como

$$u_1^4 = \sum_{j=1}^{2} w_{ji}^{34} v_j^3 = w_{11}^{34} v_1^3 + w_{21}^{34} v_2^3 = 1 \cdot e + 1 \cdot r = e + r. \tag{4.12}$$

e note que $1 \leq i \leq 2$, com i representando o $i-$ésimo neurônio na camada 4. Neste caso, por exemplo, $i = 1$ denota o neurônio cujo valor limiar é 0.5.

Analogamente, podemos descrever o potencial de ativação u_2^4 como

$$u_2^4 = \sum_{j=1}^{2} w_{ji}^{34} v_j^3 = w_{12}^{34} v_1^3 + w_{22}^{34} v_2^3 = 1 \cdot e + 1 \cdot r = e + r. \tag{4.13}$$

e a função de ativação g_4 será dada por

$$g_4 = \begin{cases} 1, & \text{se } (e+r) \geq 1.5, \\ 0, & \text{caso contrário.} \end{cases}$$

Ainda, observando a camada 4 na rede neural (o que é referente a segunda camada observando apenas a rede X_{OR}), Figura 4.13, considerando v_1^4 e v_2^4 suas entradas e $w_{11}^{45} = 1$ e $w_{21}^{45} = -2$ os pesos sinápticos das conexões entre as camada 4 e 5 temos:

$$u^5 = \sum_{j=1}^{2} w_{ji}^{45} v_j^4 = w_{11}^{45} v_1^4 + w_{21}^{45} v_2^4 = 1 \cdot v_1^4 + (-2) \cdot v_2^4 = v_1^4 - 2v_2^4. \tag{4.14}$$

Denotando

$$v_1^4 = g_{3'}(e+r) \text{ e } v_2^4 = g_4(e+r)$$

temos

$$u^5 = g_{3'}(e+r) - 2g_4(e+r).$$

Donde segue

$$g_5 = \begin{cases} 1, & \text{se } u^5 \geq 0.5, \\ 0, & \text{caso contrário.} \end{cases}$$

A aprendizagem da rede neural pode ser resumida da seguinte forma:

1. **Inicialização:** Dependendo do comprimento k da mensagem e do comprimento n da palavra do código linear, a rede neural poderá ser construída.

   Antes que a rede neural inicie o processamento da palavra recebida r, ela deve inicializar os pesos sinápticos entre as camadas.

   A inicialização dos pesos sinápticos entre as camadas $1, 2$ e 3 dependem da matriz teste de paridade H. Assim,

   $$w_{ij}^{12} = h_{ji}$$

   $$w_{ij}^{23} = \begin{cases} 1, & \text{se } h_{ij} = 1, \\ -1, & \text{se } h_{ij} = 0. \end{cases}$$

2. **Geração da síndrome:** Seja

   $$s_j^2 = \sum_{i=1}^{n} w_{ij}^{12} r_i^1.$$

   A saída da camada 2 é dada por

   $$v_j^2 = g_2(s_j^2) = \begin{cases} 1, & \text{se } s_j^2 \equiv 1 (mod 2), \\ -1, & \text{caso contrário.} \end{cases}$$

3. **Detecção do erro:** Seja

   $$s_i^3 = \sum_{j=1}^{k} w_{ji}^{23} v_j^2.$$

A saída da camada 3 é dada por

$$e_i^3 = g_3(s_i^3) = \begin{cases} 1, & \text{se } s_i^3 \geq \theta \\ 0, & \text{caso contrário.} \end{cases}$$

para $\theta = k - \frac{1}{2}$.

4. **Correção do erro:** A saída da camada 5, a palavra do código transmitida, é encontrada fazendo

$$c_i = g_5(g_{3'}(e_i + r_i) - 2 \cdot g_4(e_i + r_i)),$$

com

$$g_{3'} = \begin{cases} 1, & \text{se } (e+r) \geq 0.5, \\ 0, & \text{caso contrário;} \end{cases}$$

$$g_4 = \begin{cases} 1, & \text{se } (e+r) \geq 1.5, \\ 0, & \text{caso contrário;} \end{cases}$$

$$g_5 = \begin{cases} 1, & \text{se } (g_3(e+r) - 2 \cdot g_4(e+r)) \geq 0.5, \\ 0, & \text{caso contrário.} \end{cases}$$

Em todos os 4 casos acima, considere $1 \leq i \leq n$ e $1 \leq j \leq k$.

4.6 Exemplo do processo de decodificação por meio de redes neurais

Nesta seção, utilizamos um exemplo para demonstrar como a detecção e a correção de erros são feitas utilizando a rede neural proposta.

Seja C um código linear cuja matriz teste de paridade é dada por

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

Seja (1 0 0 1 1 0) a palavra do código transmitida[4] e $r = (1\ 0\ 0\ 1\ 1\ 1\ )$ a palavra recebida. Se somente um erro ocorreu, encontramos a palavra do código utilizando a rede proposta.

Como a matriz teste de paridade H possui 3 linhas e o comprimento da palavra do código é 6, temos $k = 3$ e $n = 6$.

1. **Geração da síndrome:** os pesos das sinapses conectadas entre as camadas 1 e 2 são dados por:

$$\begin{array}{llllll} w_{11}^{12} = 1 & w_{21}^{12} = 1 & w_{31}^{12} = 0 & w_{41}^{12} = 1 & w_{51}^{12} = 0 & w_{61}^{12} = 1 \\ w_{12}^{12} = 1 & w_{22}^{12} = 1 & w_{32}^{12} = 0 & w_{42}^{12} = 0 & w_{52}^{12} = 1 & w_{62}^{12} = 0 \\ w_{13}^{12} = 1 & w_{23}^{12} = 0 & w_{33}^{12} = 1 & w_{43}^{12} = 1 & w_{53}^{12} = 0 & w_{63}^{12} = 0. \end{array}$$

[4]Lembre-se que para este treinamento da rede, dado o vetor de entrada é necessário conhecer o vetor de saída.

As entradas da camada 1, isto é, os *bits* da palavra recebida são:

$$r_1 = 1 \quad r_2 = 0 \quad r_3 = 0 \quad r_4 = 1 \quad r_5 = 1 \quad r_6 = 1. \tag{4.15}$$

A soma ponderada (as síndromes) destas entradas são calculadas como segue

$$\begin{aligned}
s_1^2 = \sum_{i=1}^{6} w_{i1}^{12} r_i &= 1 \cdot 1 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 = 3 \\
s_2^2 = \sum_{i=1}^{6} w_{i2}^{12} r_i &= 1 \cdot 1 + 1 \cdot 0 + 0 \cdot 0 + 0 \cdot 1 + 1 \cdot 1 + 0 \cdot 1 = 2 \\
s_3^2 = \sum_{i=1}^{6} w_{i3}^{12} r_i &= 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 1 \cdot 1 + 0 \cdot 1 + 0 \cdot 1 = 2.
\end{aligned}$$

As saídas dos neurônios na camada 2 são:

$$v_1^2 = g_2(s_1^2) = 1, \;\; v_2^2 = g_2(s_2^2) = -1, \;\; v_3^2 = g_2(s_3^2) = -1.$$

2. **Detecção do erro:** os pesos sinápticos entre as camadas 2 e 3 são:

$$\begin{array}{lll}
w_{11}^{23} = 1 & w_{21}^{23} = 1 & w_{31}^{23} = 1 \\
w_{12}^{23} = 1 & w_{22}^{23} = 1 & w_{32}^{23} = -1 \\
w_{13}^{23} = -1 & w_{23}^{23} = -1 & w_{33}^{23} = 1 \\
w_{14}^{23} = 1 & w_{24}^{23} = -1 & w_{34}^{23} = 1 \\
w_{15}^{23} = -1 & w_{25}^{23} = 1 & w_{35}^{23} = -1 \\
w_{16}^{23} = 1 & w_{26}^{23} = -1 & w_{36}^{23} = -1.
\end{array}$$

As entradas dos neurônios na camada 3 são:

$$v_1^2 = g_2(s_1^2) = 1, \;\; v_2^2 = g_2(s_2^2) = -1, \;\; v_3^2 = g_2(s_3^2) = -1.$$

As somas ponderadas referentes a estas entradas são:

$$\begin{aligned}
s_1^3 = \sum_{j=1}^{3} w_{j1}^{23} v_j^2 &= w_{11}^{23} v_1^2 + w_{21}^{23} v_2^2 + w_{31}^{23} v_3^2 = 1 \cdot 1 + 1 \cdot (-1) + 1 \cdot (-1) = -1 \\
s_2^3 = \sum_{j=1}^{3} w_{j2}^{23} v_j^2 &= w_{12}^{23} v_1^2 + w_{22}^{23} v_2^2 + w_{32}^{23} v_3^2 = 1 \cdot 1 + 1 \cdot (-1) + (-1) \cdot (-1) = 1 \\
s_3^3 = \sum_{j=1}^{3} w_{j3}^{23} v_j^2 &= w_{13}^{23} v_1^2 + w_{23}^{23} v_2^2 + w_{33}^{23} v_3^2 = (-1) \cdot 1 + (-1) \cdot (-1) + 1 \cdot (-1) = -1 \\
s_4^3 = \sum_{j=1}^{3} w_{j4}^{23} v_j^2 &= w_{14}^{23} v_1^2 + w_{24}^{23} v_2^2 + w_{34}^{23} v_3^2 = 1 \cdot 1 + (-1) \cdot (-1) + 1 \cdot (-1) = 1 \\
s_5^3 = \sum_{j=1}^{3} w_{j5}^{23} v_j^2 &= w_{15}^{23} v_1^2 + w_{25}^{23} v_2^2 + w_{35}^{23} v_3^2 = (-1) \cdot 1 + 1 \cdot (-1) + (-1) \cdot (-1) = -1 \\
s_6^3 = \sum_{j=1}^{3} w_{j6}^{23} v_j^2 &= w_{16}^{23} v_1^2 + w_{26}^{23} v_2^2 + w_{36}^{23} v_3^2 = 1 \cdot 1 + (-1) \cdot (-1) + (-1) \cdot (-1) = 3,
\end{aligned}$$

com

$$e_i^3 = g_3(s_i^3) = \begin{cases} 1, & \text{se } s_i^3 \geq \theta \\ 0, & \text{caso contrário,} \end{cases}$$

para $1 \leq i \leq 6$ e $\theta = k - \dfrac{1}{2} = 3 - \dfrac{1}{2} = 2.5$, os valores de saída da camada 3 são:

$$\begin{array}{lll} e_1^3 = g_3(s_1^3) = g_3(-1) = 0 & e_2^3 = g_3(s_2^3) = g_3(1) = 0 & e_3^3 = g_3(s_3^3) = g_3(-1) = 0 \\ e_4^3 = g_3(s_4^3) = g_3(1) = 0 & e_5^3 = g_3(s_5^3) = g_3(-1) = 0 & e_6^3 = g_3(s_6^3) = g_3(3) = 1. \end{array} \tag{4.16}$$

3. **Fase de correção do erro:** considerando as funções de ativação $g_{3'}, g_4$ e g_5 definidas em (4.9), (4.10) e (4.11), respectivamente, a palavra recebida r descrita em (4.15) e o vetor erro obtido em (4.16), a saída da camada 5, a palavra do código transmitida c, é calculada pela equação

$$c_i = g_5(g_{3'}(e_i + r_i) - 2 \cdot g_4(e_i + r_i))$$

que apresenta como resultado cada *bit*, coordenada, da palavra. Assim:

$$\begin{array}{rl} c_1 = & g_5(g_{3'}(e_1 + r_1) - 2g_4(e_1 + r_1)) = g_5(g_{3'}(0 + 1) - 2g_4(0 + 1)) = g_5(1 - 2 \cdot 0) = 1 \\ c_2 = & g_5(g_{3'}(e_2 + r_2) - 2g_4(e_2 + r_2)) = g_5(g_{3'}(0 + 0) - 2g_4(0 + 0)) = g_5(0 - 2 \cdot 0) = 0 \\ c_3 = & g_5(g_{3'}(e_3 + r_3) - 2g_4(e_3 + r_3)) = g_5(g_{3'}(0 + 0) - 2g_4(0 + 0)) = g_5(0 - 2 \cdot 0) = 0 \\ c_4 = & g_5(g_{3'}(e_4 + r_4) - 2g_4(e_4 + r_4)) = g_5(g_{3'}(0 + 1) - 2g_4(0 + 1)) = g_5(1 - 2 \cdot 0) = 1 \\ c_5 = & g_5(g_{3'}(e_5 + r_5) - 2g_4(e_5 + r_5)) = g_5(g_{3'}(0 + 1) - 2g_4(0 + 1)) = g_5(1 - 2 \cdot 0) = 1 \\ c_6 = & g_5(g_{3'}(e_2 + r_2) - 2g_4(e_2 + r_2)) = g_5(g_{3'}(1 + 1) - 2g_4(1 + 1)) = g_5(1 - 2 \cdot 1) = 0. \end{array}$$

e obtemos $c = (1\ 0\ 0\ 1\ 1\ 0\ )$ como a palavra correta do código que foi transmitida.

Portanto, utilizando a rede neural proposta, a decodificação feita para códigos lineares, como no Exemplo 4.1, foi resolvida pela rede neural. Tal rede torna o decodificador de código linear rápido e fácil de ser implementado em tempo real [16].

Conclusão

A decodificação é um dos desafios da Teoria de Códigos, em se tratando tanto da complexidade de implementação quanto do desempenho da correção. Muitos trabalhos foram e têm sido desenvolvidos relacionando redes neurais e decodificação de códigos. Em vários deles o intuito é gerar melhorias que sejam significativas em relação a outros decodificadores.

O principal objetivo deste trabalho foi compreender algumas destas relações, entre redes neurais e códigos corretores de erros, bem como alguns dos métodos de decodificação descritos na literatura. Para atingir este objetivo, foi necessário estudar alguns conceitos sobre grafos e redes neurais, os quais foram apresentados conforme a necessidade para a compreensão dos resultados abordados.

Deste modo, estudamos inicialmente uma relação da rede de Hopfield com a decodificação de máxima verossimilhança em códigos lineares e, após, foi estudado um algoritmo de decodificação por meio de redes neurais para códigos corretores de erros lineares fazendo uma comparação com a decodificação por síndrome da teoria clássica. Tal método de decodificação é dependente do conhecimento da palavra do código transmitida, ou seja é necessário um conjunto de teste para o treinamento da rede.

Existem trabalhos recentes relacionando redes neurais com códigos corretores de erros, particularmente as redes de Hopfield, que propõe a junção de códigos corretores de erros com memórias associativas. Por exemplo, [13] mostra que, mesclando as abordagens de recuperação de mensagens por meio de redes de Hopfield e um algoritmo utilizado na decodificação de códigos lineares, é possível melhorar significativamente o desempenho de recuperação de informação em relação às memórias associativas.

O trabalho de [13] motivou ainda o trabalho feito por [31] que incorpora técnicas de codificação da teoria dos códigos corretores de erros em redes neurais, apresentando uma nova versão de decodificação que reduz a complexidade computacional, em relação a métodos anteriores, e é adequada para apagamentos parciais. Essas memórias associativas garantem ainda um bom desempenho na correção de erros na presença de apagamentos parciais e na recuperação de mensagens aprendidas.

Algumas das tentativas realizadas para construir decodificadores baseados em aprendizado de máquina foram em parte frustradas por limitações no algoritmo. Por exemplo, foi mostrado [26] que, mesmo em exemplos simples de códigos lineares, não era possível decodificar com sucesso vetores recebidos correspondentes às palavras do código que não foram mostradas durante o treinamento da rede, problema que foi chamado de a *maldição da dimensionalidade*.

Contudo, dois outros trabalhos recentes trouxeram contribuições a respeito deste assunto. Em [34] foi mostrado que é possível introduzir arquiteturas neurais para decodificar códigos de blocos lineares e que o problema da maldição da dimensionalidade poderia ser contornado. Em [26] foi introduzida a perda de síndrome. Tal método, projetado

para ensinar os decodificadores a produzir saídas com uma estrutura correta, apresentou melhorias no processo de decodificação e, diferentemente do que foi exemplificado neste trabalho, ele não depende do conhecimento da palavra de código transmitida.

O tema desenvolvido nesta dissertação propiciou uma base de conhecimento para futuras pesquisas nesta direção.

Referências Bibliográficas

[1] S. A. Abrantes. *Códigos corretores de erros em comunicações digitais.* Porto: FEUP Edições, 2010.

[2] D. J. Amit. *Modeling brain function: The world of attractor neural networks.* Cambridge: Cambridge University Press, 1989.

[3] M. Arenales. et al. *Pesquisa operacional para cursos de engenharia.* Rio de Janeiro: Elsevier, 2007.

[4] R. B. Bapat. *Graphs and Matrices.* New Delhi: Springer, 2010.

[5] J. A. Bondy, U. S. R. Murty. *Graph Theory.* New York: Springer, 2008.

[6] J. Bruck, M. Blaum. *Neural Networks, Error-Correcting Codes, and Polynomials over the Binary n-Cube.* IEEE Trans. Inf. Theory **35** n. 5 (1989) 976–986.

[7] J. Bruck. *On the convergence properties of the Hopfield model.* Proceedings of the IEEE **78** n. 10 (1990) 1579-1585.

[8] J. Bruck, J.W. Goodman. *A generalized convergence theorem for neural networks.* IEEE Transactions on Information Theory **34** n.5 (1988) 1089-1092.

[9] R. A. Claudio. *Aplicação de redes neurais artificiais à estimação de curtíssimo prazo do preço da energia elética.* 2014. Trabalho de conclusão de curso (Engenharia Elétrica com ênfase em Sistemas de Energia e Automação)- Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.

[10] L. G. Corrêa *Memória associativa em redes neurais realimentadas.* 2004. 137 f. Dissertação (Mestrado em Ciências da Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos.

[11] D. J. Evans, et al. *Searching sets of properties with neural networks. Parallel Comput*, **16** (1990) 279-285.

[12] G. S. Georgiou, P. Christodoulides, S. A. Kalogirou. *Implementing Artificial Neural Networks in Energy Building Applications – A Review.* In: IEEE 2018 International Energy Conference (ENERGYCON), 2018, Limassol. Disponível em https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=& arnumber=8398847.

[13] V. Gripon, C. Berrou. *A simple and efficient way to store many messages using neural cliques.*In: IEEE 2011 Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2011, Paris. Disponível em https://ieeexplore.ieee.org/document/5952106.

[14] M. Guerreiro. *Group algebras and coding theory.* São Paulo Journal of Mathematical Sciences, São Paulo, **10** (2016) 346–371. Disponível em: https://link.springer.com/article/10.1007/s40863-016-0040-x.

[15] A. C. Guyton, J. E. Hall. *Tratado de Fisiologia Médica.* 11ª Ed. Rio de Janeiro: Elsevier Ltda, 2006.

[16] A. S. Hadi. *Linear Block Code Decoder Using Neural Network.* In: 2018 IEEE International Joint Conference on Neural Networks (IEEE World Congress in Computational Intelligence), 2008, Hong Kong. Disponível em: https://ieeexplore.ieee.org/document/4633938.

[17] S. Haykin. *Redes Neurais: Princípios e prática.* Porto Alegre: Bookman, 2001.

[18] A. Hefez, M. L. T. Villela. *Códigos Corretores de Erros.* 2ª Ed. Rio de Janeiro: IMPA, 2008.

[19] A. L. Hodgkin, A. F. Huxley. *A quantitative description of membrane current and its application to conduction and excitation in nerve.* Journal of Physiology, Londres, **117** (1952) 500- 544.

[20] J. J. Hopfield. *Neural networks and physical systems with emergent collective computational abilities.* Biophysics **79** (1982) 2554-2558.

[21] M. M. Htay, S.S. Iyengar, Si-Qing Zheng. *Correcting Errors in Linear Codes with Neural Network.* In Proceedings of the 27th Southeastern Symposium in System Theory, 1995. Disponível em https://ieeexplore.ieee.org/document/390552.

[22] W. Huffmanm, V. Pless. *Fundamentals of Error-Correcting Codes.* Cambridge: Cambridge University Press, 2003.

[23] L. C. Junqueira, J. Carneiro. *J. Histologia básica: texto e atlas.* 12ª Ed. Rio de Janeiro: Guanabara Koogan, 2013.

[24] Z. L. Kovács. *Redes Neurais Artificiais: Fundamentos e Aplicações.* 4ª Ed. São Paulo: Livraria da Física, 2006.

[25] W. A. Little. *The existence of persistent states in the brain.* Mathematical Biosciences, **19** (1974) 101-120. Disponível em: https://www.sciencedirect.com/science/article/pii/0025556474900315.

[26] L. Lugosh, W. J. Gross. *Learning from the Syndrome.* In: IEEE 2018 52nd Asilomar Conference on Signals, Systems, and Computers, 2018, Pacific Grove. Disponível em https://ieeexplore.ieee.org/document/8645388.

[27] J. S. Lowe, P. G. Anderson. *Human Histology.* 4ª Ed. Philadelphia: Elsevier, Mosby, 2015.

[28] F. MacWilliams, N. Sloane. *The Theory of Error-Correcting Codes.* Amsterdam: North Holland Publishing Co., 1977.

[29] T. Matthews. *Perceptron 2: logical operations. logic and logical operations.* Computational neuroscience in excel. Disponível em < http://toritris.weebly.com/perceptron-2-logical-operations.html>. Acesso em: 12 abril 2019.

[30] W. S. McCulloch, W. Pitts. *A logical calculus of the ideas immanent in nervous activity*. Bulletin of Mathematical Biophysics **5** (1943) 115-133.

[31] A. A. Mofrad. et al. *Neural network associative memories with local coding*. In: IEEE 2015 14th Canadian Workshop on Information Theory (CWIT), 2015, St. John's. Disponível em https://ieeexplore.ieee.org/document/7255180.

[32] T. Montanari. *Histologia: Texto, atlas e roteiro de aulas práticas*. 3ª Ed. Porto Alegre: editora do Autor, 2016.

[33] L. H. A. Monteiro. *Sistemas Dinâmicos*. 2 ed. São Paulo: Livraria da Física. 2006.

[34] E. Nachmani, et al. *Deep Learning Methods for Improved Decoding of Linear Codes*. IEEE Journal of selected topics in signal processing. **12** n. 1 (2018) 119-131.

[35] L. Novak, A. Gibbons. *Hybrid Graph Theory and Network Analysis*. New York: Cambridge University Press, 1999.

[36] DQ. Nykamp. *An introduction to networks*. Disponível em <http://mathinsight.org-network-introduction>. Acesso em: 17 nov. 2018.

[37] R. Palazzo Jr, et al. *Fundamentos algébricos e geométricos dos códigos corretores de erros*. Campinas, 2006.

[38] W. W. Peterson, E. J. Weldon Jr. Error Correcting Codes. 2ª Ed. Cambridge: MIT Press, 1972.

[39] D. E. Rumelhart, K. L. McClelland. *Parallel distributed processing*. Cambridge: MIT press, 1986. v. 1.

[40] C. E. Shannon. *A Mathematical Theory of Communication*. The Bell System Technical Journal **27** n.3 (1948) 379–423.

[41] I. N. Silva, D. H. Spatti, R. A. Flauzino. *Redes Neurais Artificiais*. São Paulo: Artliber, 2010.

[42] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. 1974. Ph.D. Dissertation - Harvard University, Cambridge.