Universidade Federal do ABC
Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas
Programa de Graduação em Engenharia da Informação

Equalização Cega de Canais SIMO utilizando Autocodificadores Variacionais

Sâmya Catta Preta Sena

Santo André - SP, 02 de dezembro de 2022

Sâmya Catta Preta Sena

Equalização Cega de Canais SIMO utilizando Autocodificadores Variacionais

Texto apresentado como **Trabalho de Graduação** do Programa de Graduação em Engenharia da Informação (área de concentração: Sistemas Inteligentes), como parte dos requisitos para colação de grau.

Universidade Federal do ABC – UFABC

Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas

Programa de Graduação em Engenharia da Informação

Orientador: Prof. Dr. Claudio José Bordin Júnior

Santo André - SP

02 de dezembro de 2022

Sâmya Catta Preta Sena
Equalização Cega de Canais SIMO utilizando Autocodificadores Variacionais/ Sâmya Catta Preta Sena. – Santo André - SP, 02 de dezembro de 2022-
39 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Claudio José Bordin Júnior

Trabalho de Graduação – Universidade Federal do ABC – UFABC
Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas
Programa de Graduação em Engenharia da Informação, 02 de dezembro de 2022.

1. Equalização Cega. 2. Aprendizado de Máquina. 3. Autocodificadores Variacionais. I. Orientador: Prof. Dr. Claudio José Bordin Júnior. II. Universidade Federal do ABC. III. Centro de Engenharia e Ciências Sociais Aplicadas. IV. Equalização Cega de Canais SIMO utilizando Autocodificadores Variacionais

CDU 02:141:005.7

Sâmya Catta Preta Sena

Equalização Cega de Canais SIMO utilizando Autocodificadores Variacionais

Texto apresentado como **Trabalho de Graduação** do Programa de Graduação em Engenharia da Informação (área de concentração: Sistemas Inteligentes), como parte dos requisitos para colação de grau.

Prof. Dr. Claudio José Bordin Júnior
Orientador

Prof. Dr. André Kazuo Takahata

Prof. Dr. Marcelo Bender Perotoni

Santo André - SP
02 de dezembro de 2022

"Não é conhecimento, mas o ato de aprender, não a posse, mas o ato de chegar lá, que concede a maior satisfação". Carl Friedrich Gauss.

Agradecimentos

Agradeço ao meu orientador, Claudio José Bordin Júnior por todos os ensinamentos, conselhos e ajuda nesse período.

Aos Professores Marcelo Bender Perotoni e André Kazuo Takahata por terem aceitado avaliar este trabalho.

A minha mãe pelo encorajamento e apoio constantes.

Resumo

Deduz-se neste trabalho uma extensão do algoritmo de equalização cega baseado em autocodificadores variacionais (VAEs) de Caciularu *et al.* para canais SIMO (*Single-Input Multiple-Output*). Propõe-se ainda um novo equalizador cego linear baseado no critério do módulo constante (CMA) que realiza a busca de soluções através do método ADAM. O desempenho dos algoritmos propostos foi avaliado através de simulações numéricas, nas quais se verificou que o VAE tem um desempenho superior ao de redes neurais alternativas e que o CMA empregando o método ADAM exibe um desempenho superior ao de sua formulação tradicional.

Palavras-chaves: Equalização Cega; Critério do Módulo Constante; Autocodificadores Variacionais; Aprendizado de Máquina;

Abstract

In this work, we deduce an extension of the blind equalization algorithm based on Variational Autoencoders (VAEs) proposed by Caciularu *et al.* to SIMO (Single-Input Multiple-Output) channels. We also propose a new linear blind equalizer based on the constant modulus (CMA) criterion that searches solutions via ADAM instead of the traditional stochastic gradient descente method. Numerical simulation results indicate that the VAE outperforms alternative neural networks, and that the proposed ADAM-based CMA performs better than its traditional formulation.

Keywords: Blind Equalization; Constant Modulus Criterion; Machine Learning; Variational Autoencoders.

Sumário

Introdução

A informatização das atividades produtivas e o uso ubíquo da Internet torna necessário o desenvolvimento de sistemas de comunicação digital de capacidade cada vez mais elevada. Dada a escassez de espectro eletromagnético, o aumento de capacidade deve ser suprido preferencialmente pela melhora da eficiência espectral desses sistemas. Além disto, sistemas de comunicação digital a taxas de sinalização elevadas são fortemente impactados pela interferência intersimbólica (*Intersymbol Interference* - ISI) causada pela dispersão temporal dos sinais observada em sistemas de comunicação sem fio terrestres e a cabo. A ISI surge caso a resposta combinada do canal e dos filtros de transmissão e recepção não atenda ao Critério de Nyquist (PROAKIS; SALEHI, 2008) e pode produzir uma forte degradação de desempenho se não combatida. No domínio da frequência, um canal de comunicação com ISI possui uma resposta *seletiva* (i.e., que varia) em frequência (Figura 1).

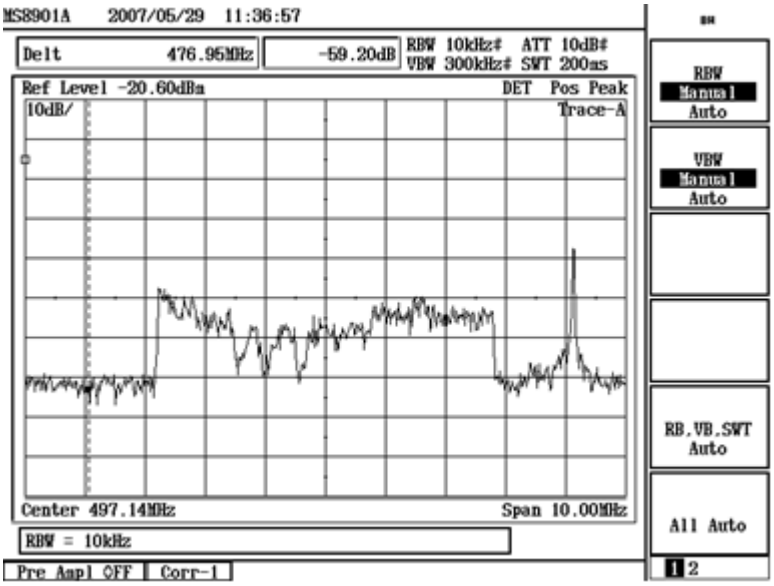

Figura 1 – Espectro do sinal recebido através de um canal de comunicação sem fio na faixa de UHF. Como o sinal transmitido possui densidade espectral de potência plana, as variações de amplitude na faixa central do gráfico são causadas pela resposta em frequência do canal. Fonte: apostila de TV Digital, Guido Stolfi.

Muitos sistemas de comunicação atuais, como redes sem fio 802.11g/n, Redes Celulares 4G e 5G (EZHILARASAN; DINAKARAN, 2017), TV Digital terrestre ISDB-T, DVB-T/T2 e a cabo DVB-C2 (WU et al., 2000)), valem-se de esquemas de modulação com múltiplas portadoras (OFDM) (RAHMATALLAH; MOHAN, 2013), que permitem mitigar os efeitos da ISI às custas de desperdício de banda devido à inserção de intervalos de guarda.

Para mitigar os efeitos da ISI em sistemas de comunicação digital a portadora única, empregam-se *equalizadores* - algoritmos para a estimação do sinal transmitido dado o sinal distorcido recebido - no processo de recepção. Há diversos tipos de algoritmos de equalização, lineares e não lineares; vide (ROMANO et al., 2018) para uma revisão sobre o assunto.

Entre os diversos tipos de equalizadores, há uma classe chamada de equalizadores não supervisionados ou cegos (BENVENISTE; GOURSAT, 1984), que recebem esta classificação porque não utilizam sinais de referência (*piloto*) conhecidos pelo receptor para a sua adaptação. Esses algoritmos, portanto, operam a partir da observação exclusiva do sinal de saída do canal. Ao dispensar a necessidade de piloto, esta classe de equalizadores pode operar de forma não cooperativa, por exemplo, para a interceptação de comunicações (HANNA; DICK; CABRIC, 2021). Além disso, ao não transmitir piloto, maximiza-se a taxa útil de dados (mantida constante a taxa de sinalização).

Dentre os equalizadores cegos lineares, classificados desta forma por operarem como filtros FIR (lineares) alimentados com as saídas do canal, as técnicas baseadas na função-custo do Módulo Constante (CMA) (SHALVI; WEINSTEIN, 1993; JOHNSON et al., 1998) são a abordagem mais comum, e já foram empregadas para o desenvolvimento de redes neurais em (YOU; HONG, 1998). Outra metodologia comumente usada é o algoritmo baseado em realimentação de decisões (*decision-directed*) (GODARD, 1980a). Dentre as técnicas não-lineares, pode-se mencionar o algoritmo de (GHOSH; WEBER, 1992), que calcula iterativamente uma aproximação da estimativa de máxima verossimilhança (ML) dos parâmetros do canal. Esta técnica pode ser combinada à decodificação utilizando o algoritmo de Maximização de Expectativas (*expectation maximization* - EM) (WANG; CHEN, 2001). No entanto, estes métodos se tornam proibitivos para canais com número elevado de parâmetros em virtude de sua complexidade computacional, por necessitarem da aplicação de algoritmos como o de Viterbi (FORNEY, 1973; SHLEZINGER et al., 2020).

0.1 *Deep Learning*

Técnicas (não-lineares) de aprendizagem de máquina têm sido amplamente empregadas na solução de problemas de processamento de sinais (LITTLE, 2019). Dentre elas, destacam-se as redes neurais com múltiplas camadas (*Deep Learning* - DL) (LECUN; BENGIO; HINTON, 2015), em diversas formulações. Os autocodificadores variacionais (*Variational Autoencoders* - VAEs) (KINGMA; WELLING, 2013) são redes neurais com múltiplas camadas que visam reconstruir um sinal impondo estrutura ao mesmo. Os VAEs podem ser empregados para compressão de sinais ou redução de ruído, dentre outras aplicações (DOERSCH, 2016).

As técnicas de DL têm encontrado aplicações em sistemas de comunicação. Em (NACHMANI; BE'ERY; BURSHTEIN, 2016), foi utilizado DL para a decodificação de códigos corretores de erros lineares. Em (O'SHEA; HOYDIS, 2017), por sua vez, propõe-se uma formulação alternativa para sistemas de comunicação, inspirada em DL, como uma tarefa de reconstrução de sinais que busca otimizar conjuntamente componentes do transmissor e do receptor. Já em (SAMUEL; DISKIN; WIESEL, 2017), técnicas de DL foram utilizadas para detecção em sistemas MIMO (*Multiple-Input Multiple-Output*).

Métodos de DL também têm sido empregados em problemas de equalização cega. O uso de VAEs para problemas de equalização cega é novo (DING; LI, 2018; CACIULARU; BURSHTEIN, 2018). Outros tipos de redes neurais, porém, já foram empregados para este fim. Em (DAI, 2001), por exemplo, propõe-se o uso de uma rede neural convolucional para a equalização cega de um canal de comunicação utilizando uma função custo inspirada na do algoritmo CMA (GODARD, 1980b).

0.2 Contribuições e Organização do Texto

Neste texto, propõe-se a extensão de um dos algoritmos propostos em (CACIULARU; BURSHTEIN, 2020) para canais SIMO (*Single-Input Multiple-Output*). O novo método é deduzido sob a hipótese de que o ruído aditivo seja independente nos múltiplos receptores. Propõe-se, ainda, um novo equalizador cego linear baseado no critério do módulo constante que realiza a busca de soluções através do método ADAM (KINGMA; BA, 2014) ao invés do usual método de gradiente estocástico descendente. O desempenho dos métodos propostos é, então, avaliado através de simulações numéricas utilizando programas escritos em Python.

Dentre as vantagens dos algoritmos propostos, podem-se mencionar: taxas médias de erro mais baixas que as observadas para algoritmos alternativos, aquisição dos parâmetros do canal de menor latência.

O texto a seguir está organizado da seguinte forma: no Capítulo 1, descreve-se o modelo de sinal considerado e algoritmos clássicos para equalização adaptativa. No Capítulo 2, é feita uma revisão de técnicas de aprendizado profundo. No Capítulo 3, por sua vez, apresenta-se a dedução da extensão do algoritmo de (CACIULARU; BURSHTEIN, 2020) para canais SIMO bem como uma versão do algoritmo anterior com complexidade computacional reduzida. O resultado de simulações numéricas é relatado no Capítulo 4. Finalizando, o Capítulo 5 reúne as conclusões do estudo realizado e propostas para a sua continuação.

1 Formulação do Problema

Seja x_n uma sequência de símbolos transmitida através de um sistema de modulação digital linear. Supondo que o receptor tenha adquirido corretamente o sincronismo de portadora, pode-se adotar o seguinte modelo em banda base para o sinal recebido $y(t)$:

$$y(t) = \sum_{n=-\infty}^{\infty} x_n \, h(t - nT) + v(t), \tag{1.1}$$

em que $h(t)$ é a resposta ao impulso combinada dos filtros de transmissão e de recepção e do canal de transmissão, T é o intervalo entre símbolos e $v(t)$ é o ruído aditivo (Figura 2).

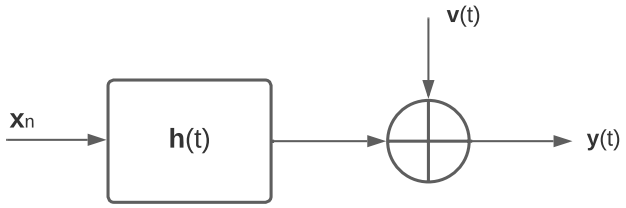

Figura 2 – Modelamento do Sinal y(t). Fonte: autoria própria.

Supõe-se neste trabalho que o tempo de coerência dos canais de transmissão seja muito maior que o intervalo de sinalização. Assim, pode-se supor que a resposta ao impulso $h(t)$ seja invariante no tempo durante o intervalo de interesse.

Amostrando-se o sinal $y(t)$ uniformemente com período $\tau = T/R$, $R \in \mathbb{N}$, obtém-se a sequência $y_n \triangleq y(n\tau)$. Esta sequência pode ser expressa em função dos demais parâmetros como

$$\begin{aligned} y_n = y(n\tau) &= \left[\sum_{k=-\infty}^{\infty} x_k h(n\tau - kT) \right] + v(n\tau) \\ &= \left[\sum_{k=-\infty}^{\infty} x_k h\left(\left(\frac{n}{R} - k \right) T \right) \right] + v\left(\frac{n}{R} T \right). \end{aligned} \tag{1.2}$$

Para fatores de sobreamostragem $R > 1$, (1.2) não representa uma operação de convolução convencional. Uma formulação alternativa pode ser obtida reescrevendo (1.2) em função das seguintes sequências:

$$y_{n,r} \triangleq y\left(\left(n + \frac{r}{R} \right) T \right), \tag{1.3}$$

$$h_{n,r} \triangleq h\left(\left(n + \frac{r}{R} \right) T \right), \tag{1.4}$$

$$v_{n,r} \triangleq v\left(\left(n + \frac{r}{R} \right) T \right). \tag{1.5}$$

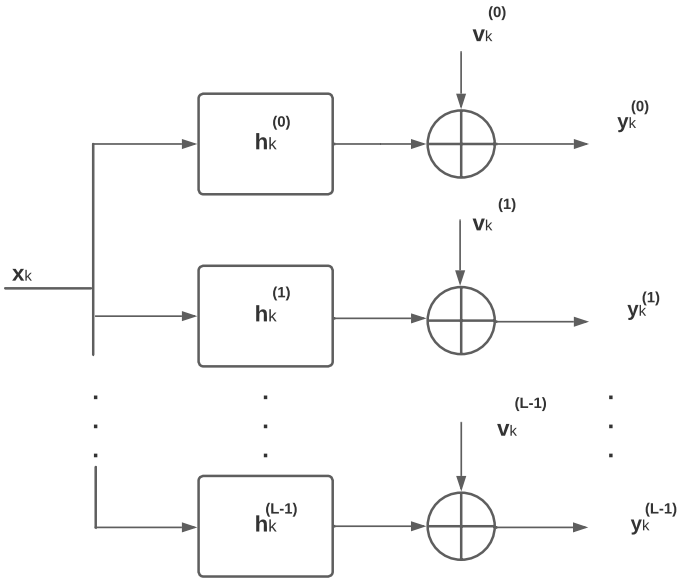

Figura 3 – Modelo SIMO de tempo discreto resultante da amostragem do sinal $y(t)$ a uma taxa $\tau = T/R$. Fonte: autoria própria.

Substituindo (1.3)-(1.5) em (1.2), obtém-se a seguinte expressão para as subsequências $y_{n,r}, 0 \leq r < R$:

$$y_{n,r} = \sum_{k=-\infty}^{\infty} x_k h_{n-k,r} + v_{n,r}. \tag{1.6}$$

Supondo que a função de transferência combinada $h(t)$ seja causal e nula para $t \geq LT$, obtém-se finalmente que

$$y_{n,r} = \sum_{k=0}^{L-1} x_k h_{n-k,r} + v_{n,r}. \tag{1.7}$$

A Equação (1.7) pode ser interpretada como um canal SIMO, com entrada x_n e saídas $y_{n,r}$, $0 \leq r < R$ (Figura 3). Saliente-se que a formulação das Equações (1.2)-(1.6) não é a única forma possível de se obter um canal SIMO num sistema de comunicação digital. Um modelo como o da Equação (1.7) pode ser obtido sem o uso de sobreamostragem através de R receptores; neste caso $h_{n,r}$ representaria a resposta ao impulso equivalente do canal entre o transmissor e o receptor de índice r. É possível também obter um modelo equivalente através de outras técnicas, como diversidade em frequência ou no tempo (PROAKIS; SALEHI, 2008).

Neste texto, supõe-se que $x_k \in \{-1,\ 1\}$ seja uma sequência de símbolos BPSK equiprováveis e i.i.d. (independentes e identicamente distribuídos), e que o $v_{n,r}$ seja um processo Gaussiano i.i.d. de média nula, variância σ_r^2, independente de x_k e de[1] $v_{n,s}$, $s \neq r$.

[1] Observe que esta suposição não é obedecida caso o modelo SIMO seja obtido através da sobreamostragem do sinal recebido se o filtro receptor tiver a largura de faixa mínima (igual a $1/T$ Hz) pois, neste caso, a contribuição do ruido aditivo após o filtro de recepção não possui espectro plano.

Os algoritmos analisados neste texto operam em *batch*, processando de cada vez N amostras dos sinais recebidos. Assim, utilizando-se a notação

$$\begin{cases} \boldsymbol{x} & \triangleq \{x_0, x_1, \ldots, x_{N-1}\}, \\ \boldsymbol{v}_r & \triangleq \{v_{0,r}, v_{1,r}, \ldots, v_{N-1,r}\}, \\ \boldsymbol{y}_r & \triangleq \{y_{0,r}, y_{1,r}, \ldots, y_{N-1,r}\}, \\ \boldsymbol{y} & \triangleq \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_R\}, \\ \boldsymbol{h}_r & \triangleq \{h_{0,r}, h_{1,r}, \ldots, h_{L-1,r}\}, \end{cases}$$

pode-se escrever $\boldsymbol{y}_r = \boldsymbol{x} * \boldsymbol{h}_r + \boldsymbol{v}_r$, em que $*$ denota a operação de convolução.

O objetivo dos algoritmos de equalização cega analisados a seguir é estimar $\boldsymbol{x}$ dadas somente as observações $\boldsymbol{y}$.

1.1 Equalização Adaptativa de Canais SIMO

Os equalizadores adaptativos lineares (SAYED, 2011) determinam filtros FIR que, colocados em série com o canal de comunicação, aproximadamente invertem os seus efeitos, combatendo a ISI. Nas seções a seguir, descrevem-se os algoritmos LMS (*Least Mean Squares*), RLS (*Recursive Least Squares*) e CMA (*Constant Modulus Algorithm*) utilizados neste trabalho.

1.1.1 O algoritmo LMS

O algoritmo LMS (SAYED, 2011) aproxima a solução $\mathbf{g}_{MMSE}$ de um problema de estimação da forma

$$\mathbf{g}_{MMSE} = \arg\min_{\mathbf{g}} E\left\|x_{n-d} - \mathbf{g}^T\mathbf{Y}_n\right\|^2, \tag{1.8}$$

em que x_n é o sinal de referência, $d \geq 0$ é um atraso (arbitrário) e $\mathbf{Y}_n$ é um vetor denominado *regressor*.

Ao empregar o algoritmo LMS num problema de equalização SIMO, x_n é a própria sequência de símbolos transmitida, que deve ser conhecida pelo receptor durante uma porcentagem do tempo (constituindo os chamado *símbolos piloto*), $\mathbf{g}$ é um vetor que coleciona os *taps* do filtro equalizador e $\mathbf{Y}_n$ é um vetor que contém as últimas L (ordem do filtro) amostras do sinal recebido por cada um dos subcanais:

$$\mathbf{Y}_n = \begin{bmatrix} \mathbf{Y}_{n,1} \\ \mathbf{Y}_{n,2} \\ \vdots \\ \mathbf{Y}_{n,R} \end{bmatrix}, \qquad \mathbf{Y}_{n,r} = \begin{bmatrix} \mathbf{y}_{n,r} \\ \mathbf{y}_{n-1,r} \\ \vdots \\ \mathbf{y}_{n-L+1,r} \end{bmatrix}. \tag{1.9}$$

A formulação do algoritmo LMS é bastante simples, resumindo-se à expressão

$$\mathbf{g}_n = \mathbf{g}_{n-1} + \mu \mathbf{Y}_n \left(x_{n-d} - \mathbf{g}_{n-1}^T \mathbf{Y}_n \right), \tag{1.10}$$

em que $\mu \ll 1$ é uma constante de adaptação e $\boldsymbol{g}_0 = \mathbf{0}$.

Mostra-se (SAYED, 2011) que quanto maior o valor de μ, mais rápida é a convergência do algoritmo para o seu estado de regime e maior a capacidade do algoritmo de rastrear variações na solução ótima (caso o canal seja variante no tempo). Ocorre porém, que $\lim_{n\to\infty} E\|\mathbf{g}_n - \mathbf{g}_{MMSE}\|^2$ cresce em função de μ. Além disso, valores muitos altos para μ provocam a divergência do algoritmo. Assim, a escolha do valor de μ estabelece um compromisso entre qualidade da solução (menor μ) e a velocidade de convergência para a solução ótima (maior μ).

1.1.2 O algoritmo RLS

O algoritmo RLS (SAYED, 2011) determina recursivamente o filtro $\mathbf{g}_n$ que minimiza o custo

$$C(\mathbf{g}_n) = \sum_{i=0}^{n} \lambda^{n-i} \left\| x_{n-d} - \mathbf{g}_n^T \mathbf{Y}_n \right\|^2, \tag{1.11}$$

em que $0 \ll \lambda < 1$ é o chamado *fator de esquecimento*.

Observe que, diferentemente do algoritmo LMS, o custo minimizado pelo algoritmo RLS é determinístico e que, quanto menor o valor de λ, menor a influência de amostras passadas no custo.

Há diversas implementações computacionais possíveis para o algoritmo RLS, que são equivalentes do ponto de vista matemático (mas não caso os cálculos sejam efetuados em precisão finita). Neste trabalho, considerou-se a implementação clássica do mesmo, com complexidade computacional quadrática na ordem, cujo equacionamento é dado abaixo:

$$\boldsymbol{K}_n = \boldsymbol{P_{n-1}} \boldsymbol{Y}_n / \left(\lambda + \boldsymbol{Y}_n^T \boldsymbol{P_{n-1}} \boldsymbol{Y}_n \right), \tag{1.12}$$

$$\boldsymbol{P}_n = \lambda^{-1} \left[\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{Y}_n^T \right] \boldsymbol{P}_{n-1}, \tag{1.13}$$

$$\boldsymbol{g}_n = \boldsymbol{g}_{n-1} + \boldsymbol{K}_n \left(x_{n-d} - \mathbf{g}_{n-1}^T \mathbf{Y}_n \right), \tag{1.14}$$

com $\boldsymbol{P}_0 = \delta \boldsymbol{I}$, $\delta \gg 1$, e $\boldsymbol{g}_0 = \mathbf{0}$.

O vetor $\boldsymbol{K}_n$, da mesma dimensão do regressor, é conhecido por *ganho de Kalman* e a matriz $\boldsymbol{P}_n$ é o inverso de uma estimativa amostral (ponderada exponencialmente) da matriz de covariância do regressor.

1.1.3 O algoritmo do módulo constante (CMA - *Constant Modulus Algorithm*)

O CMA (SAYED, 2011) determina recursivamente um filtro $\boldsymbol{g}_n$ que minimiza o custo

$$C(\boldsymbol{g}_n) = E \left\| 1 - \left| \mathbf{g}_n^T \mathbf{Y}_n \right|^2 \right\|^2. \tag{1.15}$$

Diferentemente dos algoritmos LMS e RLS, o CMA é dito *cego*, por operar sem o conhecimento do sinal de referência x_n. Ao invés disso, o algoritmo se vale da propriedade das constelações BPSK e QPSK (PROAKIS; SALEHI, 2008) de ter todos os símbolos com módulo unitário, determinando um filtro $\mathbf{g}_n$ que busca restaurar esta propriedade.

O CMA otimiza o custo da Equação 1.15 através do método de gradiente estocástico (SAYED, 2011), resultando na recursão

$$z_n = x_{n-d} - \mathbf{g}_{n-1}^T \mathbf{Y}_n, \tag{1.16}$$

$$\mathbf{g}_n = \mathbf{g}_{n-1} + \nu \mathbf{Y}_n z_n (1 - z_n^2). \tag{1.17}$$

Diferentemente do algoritmo LMS, o CMA não pode ser inicializado com $\mathbf{g}_n = \mathbf{0}$, pois, neste caso, não haveria adaptação. O valor de ν deve ser suficientemente próximo de zero para evitar a divergência do algoritmo.

Infelizmente, o custo dado pela Equação 1.15 não é uma função convexa de $\mathbf{g}_n$, o que pode provocar a convergência do algoritmo baseado em gradiente estocástico para mínimos locais correspondentes a soluções insatisfatórias. Uma alternativa proposta neste trabalho contornar para este problema é utilizar o otimizador ADAM (*Adaptive Moment Estimation*) (KINGMA; BA, 2014) na minimização de $C(\boldsymbol{g}_n)$. O algoritmo ADAM é uma generalização do método de gradiente descendente (SAYED, 2011), que calcula recursivamente a média e o segundo momento do gradiente do custo a otimizar, de acordo com

$$m_i^{(t+1)} = \beta_1 m_i^{(t)} + (1 - \beta_1) \nabla_i C^{(t)}, \tag{1.18}$$

$$u_i^{(t+1)} = \beta_2 u_i^{(t)} + (1 - \beta_2) (\nabla_i C^{(t)})^2, \tag{1.19}$$

$$\hat{m}_i = \frac{m_i^{(t+1)}}{1 - \beta_1}, \tag{1.20}$$

$$\hat{u}_i = \frac{u_i^{(t+1)}}{1 - \beta_2}, \tag{1.21}$$

$$g_i^{(t+1)} = g_i^{(t)} - \eta \frac{\hat{m}_i}{\sqrt{\hat{u}_i} + \epsilon}, \tag{1.22}$$

em que $\epsilon \ll 1$ (em geral, 10^{-8}), $\beta_1 = 0,9$ e $\beta_2 = 0,999$ são fatores de esquecimento para o primeiro e segundo momento dos gradientes, respectivamente, $\nabla_i C^{(t)}$ denota o $i-$ésimo elemento do gradiente (instantâneo) de $C(\boldsymbol{g}_n)$, dado por $-2\boldsymbol{Y}_n z_n (1 - z_n^2)$, g_i o $i-$ésimo elemento do vetor $\boldsymbol{g}_n$ e t denota o índice da iteração de treinamento (em geral, diferente de n).

2 Estruturas para *Deep Learning*

Neste capítulo descrevem-se as chamadas redes neurais artificiais e, em seguida, os autocodificadores, um tipo específico de rede neural utilizado em problemas de aprendizado não supervisionado.

2.1 Redes Neurais

Redes neurais (RNs) são sistemas de processamento de informação distribuídos de funcionamento supostamente similar ao do cérebro humano, que são capazes de aprender e se auto-organizar. Elas são compostas por unidades de processamento simples chamadas de neurônios que são interconectados de modo a formar uma rede capaz de executar tarefas computacionais complexas. As RNs possuem várias propriedades que as tornam muito atrativas para aplicações em comunicações digitais, das quais pode-se citar: arquitetura distribuída paralela, processamento adaptativo, capacidade de auto-organização e de aproximação universal e possibilidade de implementação eficiente em hardware.

As redes neurais realizam duas atividades principais: aprendizagem e generalização (TSOUKALAS; UHRIG, 1996). Na aprendizagem, ocorre o processo de adaptação dos parâmetros de conexão para minimizar uma função de perda dado um vetor de entrada. O procedimento é interrompido assim que o critério de aprendizagem previamente determinado tenha sido satisfeito. A generalização, por sua vez, é o processo de aceitar um novo vetor de entrada e, a partir dos pesos atribuídos à rede, produzir uma resposta de saída. Existem dois principais tipos de aprendizado: supervisionado e não supervisionado.

No aprendizado supervisionado (Figura 4), a rede é treinada por uma sequência de entrada[1] x_n para aproximar uma resposta desejada d_n. Assim, a rede tem de minimizar um erro e_n, definido como

$$e_n = d_n - y_n, \tag{2.1}$$

em que d_n denota a resposta desejada e y_n a saída de rede.

[1] Neste capítulo, o índice subscrito n não se refere à necessariamente dimensão temporal.

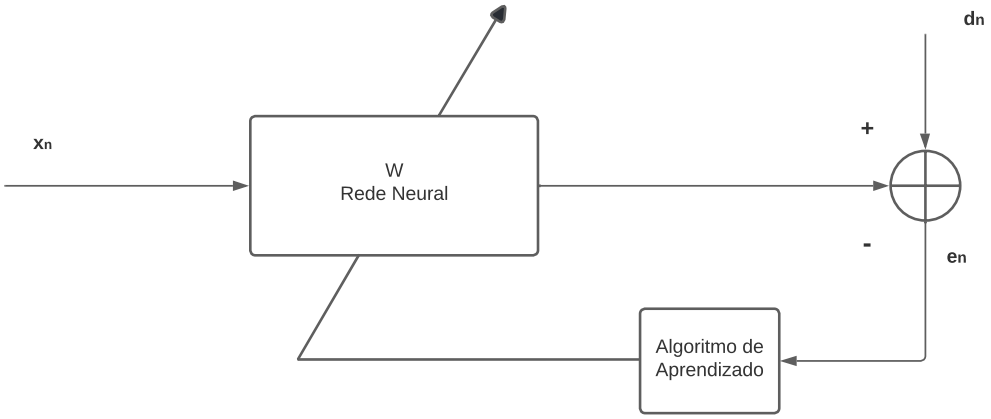

Figura 4 – Aprendizado Supervisionado. Fonte: autoria própria.

No aprendizado não supervisionado a rede não dispõe da resposta desejada. Com o sinal de entrada x_n aplicado, a resposta dependerá de capacidade da rede de se auto-organizar para extrair características relevantes desse sinal.

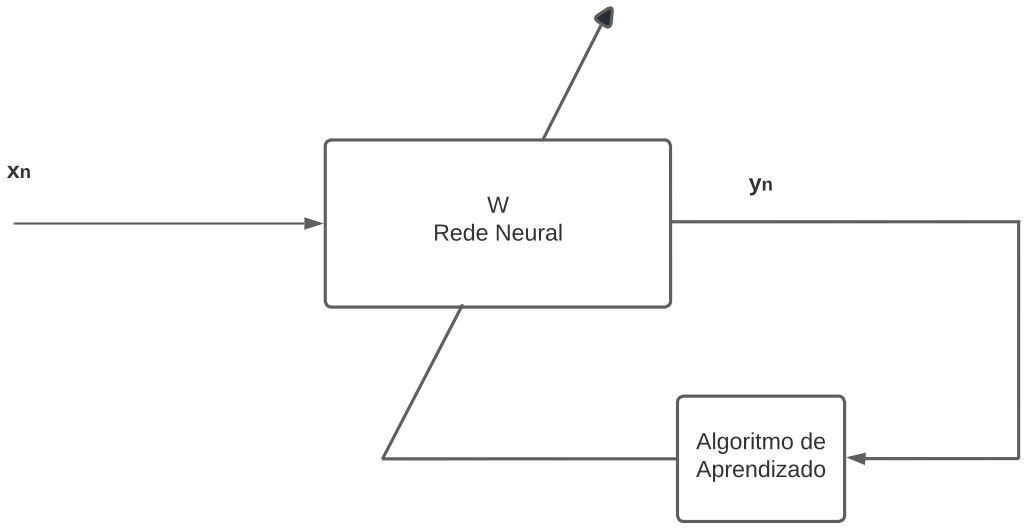

Figura 5 – Aprendizado não Supervisionado. Fonte: autoria própria.

2.1.1 Perceptron

A unidade básica de uma rede neural, o neurônio, é composto por uma função agregadora, mais comumente um combinador linear, e uma função de ativação.

O combinador linear é formado por um conjunto de n *sinapses*. A sinapse i do neurônio j é caracterizada pelos pesos W_{ij} que se aplicam às entradas x_i, $0 \leq i \leq n$, gerando a saída y_j como

$$y_j = f\left(\sum_{i=1}^{n} x_i W_{ij} + b_j\right), \tag{2.2}$$

em que $f(\cdot)$ é a função de ativação e b_j o termo de polarização (*bias*).

Os pesos W_{ij} e o termo de polarização b_j são os parâmetros livres do neurônio j. Este modelo de neurônio é chamado de *perceptron* (Figura 7).

Dentre as funções de ativação mais utilizadas, destacam-se a função ReLU (*Rectified Linear Unit*), definida como

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases},$$

e a função *sigmoide* (Figura 6), definida como

$$f(x) = \frac{1}{1 + e^{-x}}.$$

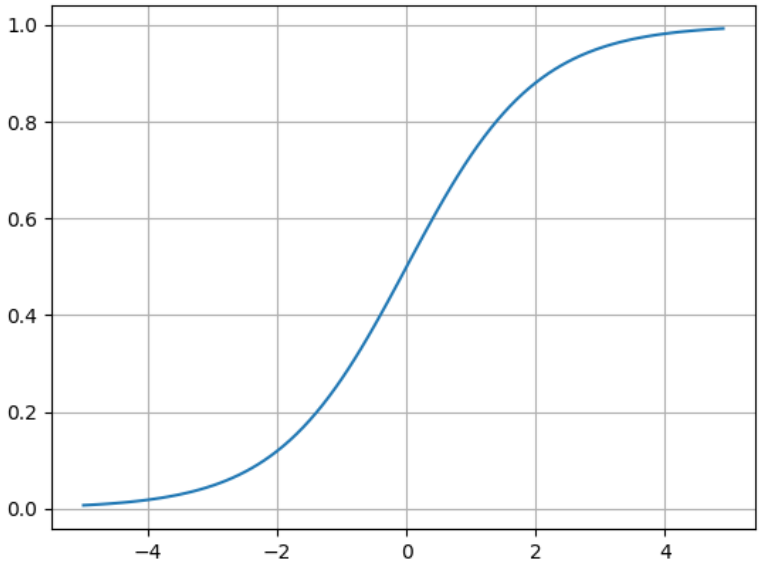

Figura 6 – Função Sigmoide $f(x) = \frac{1}{1+\exp(-x)}$. Fonte: autoria própria.

A referência (HAYKIN, 2001) descreve outras possíveis funções de ativação que podem ser utilizadas a depender do problema em questão.

2.1.2 Multilayer Perceptron

Dentre as arquiteturas mais utilizadas de redes neurais, destaca-se o *Multi-Layer Perceptron* (MLP), composto por perceptrons conectados uns aos outros em múltiplas camadas. Classicamente, um MLP possui ao menos 3 camadas: a de entrada, a intermediária ou *oculta* e a de saída.

A saída de cada neurônio é chamada de *ativação*. As ativações em uma camada determinam as ativações das camadas seguintes, num mecanismo conhecido como *feedfoward*.

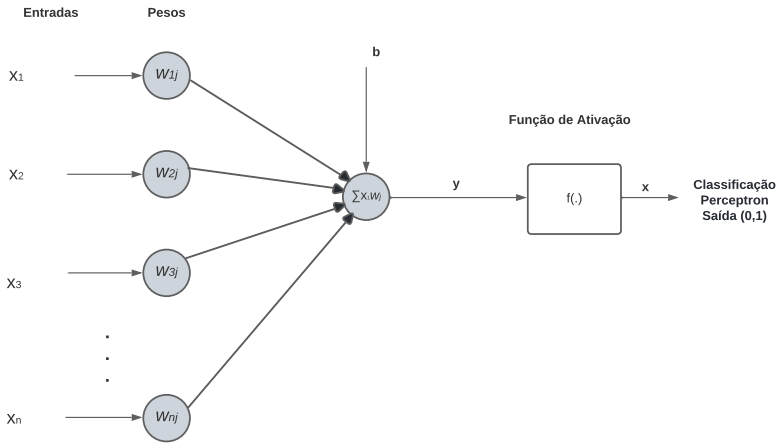

Figura 7 – Modelo de Neurônio Artificial. Fonte: autoria própria.

Por este mecanismo, as entradas e as saídas das rede são, respectivamente, as entradas da primeira camada e as saídas da última camada.

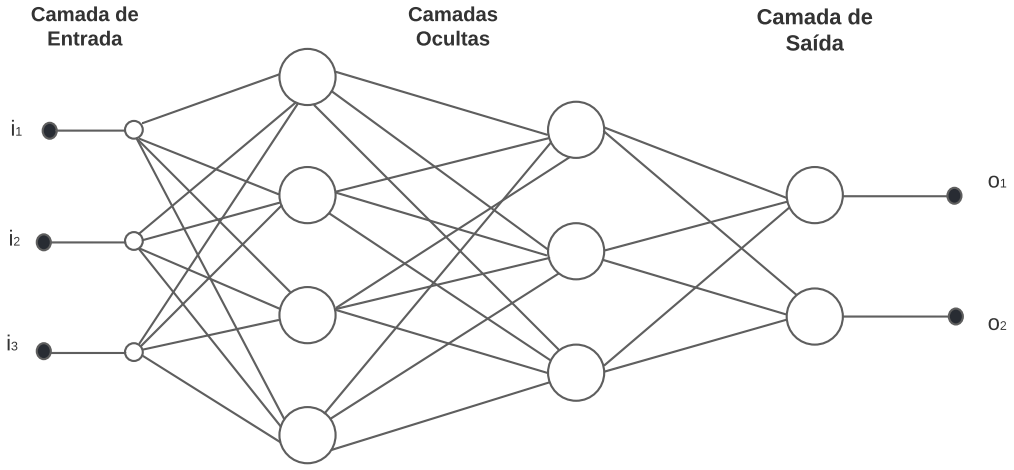

Figura 8 – Multilayer Perceptron: A rede possui 3 sinais de entrada, 4 neurônios na primeira camada oculta, 3 neurônios na segunda camada oculta, 2 neurônios na camada de saída e 2 sinais de saída. Fonte: autoria própria.

A estrutura de um MLP é representada na Figura 8. Num MLP, a camada de entrada não desempenha nenhum papel computacional, apenas servindo para inserir os dados do vetor de entrada na rede. Na Figura 8, $[i_1\ i_2\ i_3]$ é um vetor que contém as entradas e $[O_1\ O_2]$ as saídas da rede. Um MLP pode ser utilizado como base para uma estrutura de *Deep Learning*, bastando que o mesmo contenha um número grande de camadas (LECUN; BENGIO; HINTON, 2015).

2.2 Autocodificadores

Um autocodificador (PINAYA et al., 2020) é um tipo específico de rede neural, projetado especificamente para criar uma representação compactada das entradas, chamada de *variáveis latentes*. A partir das variáveis latentes, o autocodificador busca *reconstruir* as entradas produzindo o menor erro possível de acordo com alguma métrica, como erro quadrático ou erro absoluto. O procedimento de geração das variáveis latentes é chamado de *codificação*, e o de reconstrução das entradas de *decodificação*.

2.2.1 Modelos de Variáveis Latentes

Em aprendizado de máquina existem dois tipos de modelagem: generativa e discriminativa. Enquanto na modelagem discriminativa busca-se determinar um preditor a partir das observações, na modelagem generativa busca-se inferir a distribuição conjunta dos dados $\boldsymbol{X}$ observados, em algum espaço potencialmente de alta dimensão (KINGMA; WELLING et al., 2019). O modelo generativo possibilita, então, a obtenção de previsões simulando-se desta distribuição conjunta.

Embora os modelos generativos possam aprender de forma eficiente, eles tendem a sobreajustar o conjunto de dados, o que é menos comum com o modelo discriminativo (KINGMA; WELLING et al., 2019). Entretanto, os métodos discriminativos podem ser insatisfatórios quando a quantidade de dados for muito grande, sendo conveniente, neste caso, a sua substituição por modelos generativos.

Ao treinar um modelo generativo, quanto mais complexas forem as dependências entre os dados, mais difícil será o treinamento. Para tratar a questão de maneira mais concreta, tome-se como exemplo um pretenso gerador de algarismos aleatórios que funcione acionando segmentos de um *display* numérico no formato de sete segmentos (Figura 9): o acionamento aleatório de 7 segmentos pode produzir 128 padrões distintos, dos quais apenas 10 são algarismos válidos. Isto implica que seria mais efetivo se o modelo decidisse primeiro qual algarismo gerar ao invés de acionar segmentos do *display* e testar a validade do resultado. Neste caso, o algarismo a ser gerado constitui a *variável latente*. Ao proceder desta forma, o modelo primeiro amostraria aleatoriamente um valor de dígito do conjunto $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ e, em seguida, acionaria os segmentos correspondentes ao algarismo escolhido. A variável é chamada de "latente" por não ser necessariamente conhecida.

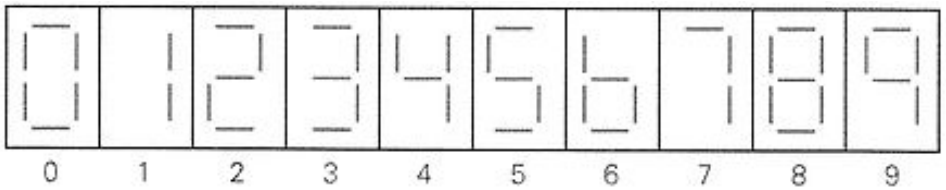

Figura 9 – Display de sete segmentos (Fonte: Wikipedia).

Para que um modelo seja de fato representativo é necessário que, para cada ponto de dados no conjunto, existam uma ou mais configurações das variáveis latentes que façam com que o modelo gere algo próximo ou semelhante a ele. Formalmente, considere um vetor de variáveis latentes $\boldsymbol{z}$ definido em um espaço de alta dimensão $\boldsymbol{Z}$ no qual se pode amostrar de alguma função densidade de probabilidade (f.d.p.) $p(\boldsymbol{z})$. Supondo que haja uma família de variáveis determinísticas de funções $f(\boldsymbol{z};\boldsymbol{\theta})$ parametrizadas por um vetor $\boldsymbol{\theta}$ em algum espaço $\boldsymbol{\Theta}$, $f : \boldsymbol{Z} \times \boldsymbol{\Theta} \rightarrow \boldsymbol{X}$, em que f é determinístico, $\boldsymbol{z}$ é aleatório e $\boldsymbol{\theta}$ fixo, o que torna $f(\boldsymbol{z};\boldsymbol{\theta})$ uma variável aleatória no espaço $\boldsymbol{X}$, deseja-se otimizar $\boldsymbol{\theta}$ de modo que se possa amostrar $\boldsymbol{z}$ de $p(\boldsymbol{z})$ e, com alta probabilidade, $f(\boldsymbol{z};\boldsymbol{\theta})$ será similar a algum $\boldsymbol{x}$ do conjunto de dados. Em outras palavras, busca-se maximizar a probabilidade de cada $\boldsymbol{x}$ do conjunto de treinamento de acordo com:

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})p(\boldsymbol{z})d\boldsymbol{z}. \tag{2.3}$$

Na Eq. (2.3), $f(\boldsymbol{z};\boldsymbol{\theta})$ foi substituído por $p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})$, de modo a explicitar a dependência entre de $\boldsymbol{x}$ em $\boldsymbol{z}$ se valendo da lei da probabilidade total. A estrutura tratada se vale do critério de máxima verossimilhança que garante que, se o modelo provavelmente produzir amostras do conjunto de treinamento, produzirá também amostras semelhantes e, improvavelmente, amostras diferentes. Frequentemente, escolhe-se uma f.d.p. gaussiana, ou seja, $p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}|f(\boldsymbol{z};\boldsymbol{\theta}), \sigma^2\boldsymbol{I})$, em que $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denota uma f.d.p. Gaussiana multivariada com vetor de média $\boldsymbol{\mu}$ e matriz de covariância $\boldsymbol{\Sigma}$, e $\boldsymbol{I}$ denota a matriz identidade.

De modo geral, no início do treinamento, o modelo não produz saídas idênticas a qualquer $\boldsymbol{x}$ específico. Ao utilizar uma f.d.p. Gaussiana, pode-se valer do uso de técnicas de otimização como o gradiente descendente para maximizar $p(\boldsymbol{x})$ fazendo $f(\boldsymbol{z};\boldsymbol{\theta})$ aproximar $\boldsymbol{x}$ para algum $\boldsymbol{z}$, ou seja, gradualmente tornando os dados de treinamento mais prováveis sob o modelo generativo. Isso não seria possível se $p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})$ fosse um Delta de Dirac, como seria se se usasse $\boldsymbol{x} = f(\boldsymbol{z};\boldsymbol{\theta})$ de forma determinística. A f.d.p. $p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})$ não precisa necessariamente ser Gaussiana. Por exemplo, se $\boldsymbol{x}$ for binário, $p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})$ pode ser Bernoulli, parametrizada por $f(\boldsymbol{z};\boldsymbol{\theta})$. A propriedade importante é que $p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})$ seja contínua em $\boldsymbol{\theta}$ e possa ser calculada (KINGMA; WELLING et al., 2019).

2.2.2 Autocodificadores Variacionais

O autocodificador variacional (VAE) (KINGMA; WELLING et al., 2019) é um tipo específico de autocodificador no qual o codificador produz uma aproximação da densidade *a posteriori* das *variáveis latentes* condicionada à entrada observada. Os parâmetros desta densidade são obtidos como aqueles que maximizam o limitante inferior de evidência (também conhecido como limitante inferior variacional) (KINGMA; WEL-

LING, 2013) ou utilizando-se o algoritmo de maximização de expectativas (*Expectation Maximization*) (BALESTRIERO; PARIS; BARANIUK, 2020).

Segundo (KINGMA; WELLING et al., 2019), "*a busca por fatores de variação nos dados desembaraçados, semanticamente significativos, estatisticamente independentes e causais é geralmente conhecido como aprendizado de representação não supervisionado, e VAEs tem sido amplamente empregados para esse propósito*".

Num VAE, o decodificador pode ser implementado como num autocodificador comum, usando como entrada uma amostra gerada a partir da densidade *a posteriori* obtida pelo codificador. Em algumas implementações de VAEs (CACIULARU; BURSHTEIN, 2020), o decodificador não é implementado diretamente, sendo ao invés disso calculadas estatísticas do erro de reconstrução.

A base matemática dos VAE's pouco se assemelha a dos autocodificadores clássicos, como os autocodificadores esparsos (NG et al., 2011) ou os autocodificadores eliminadores de ruído (*denoising autoencoders*) (VINCENT et al., 2008). A única razão de os VAE's serem chamados de tal forma é por possuírem um codificador e um decodificador.

3 Autocodificador Variacional para Equalização Cega de Canais SIMO

Neste capítulo é descrita uma generalização do algoritmo VAE (CACIULARU; BURSHTEIN, 2020) para canais SIMO.

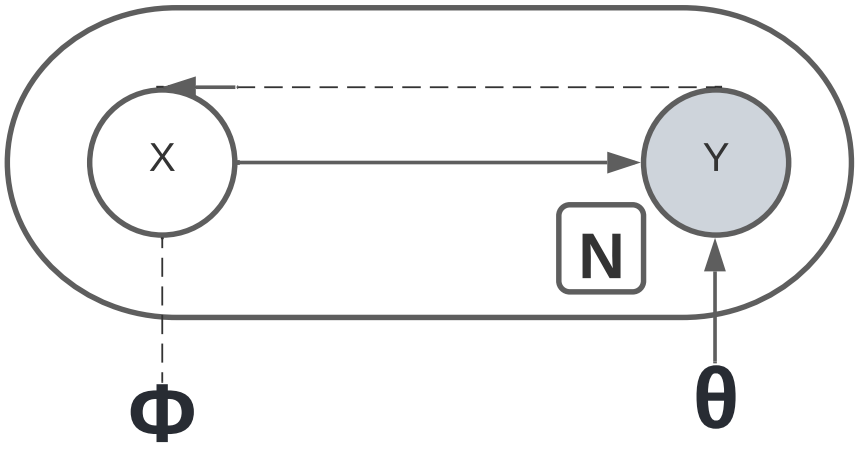

Figura 10 – Modelo Gráfico Direcional para Canal de Comunicação e para a Estrutura da Rede para Inferência. O modelo generativo $p(\boldsymbol{x})p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ é denotado pela linha sólida. A aproximação variacional $q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})$ para $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y})$ é denotada pela linha tracejada. (Adaptado de (CACIULARU; BURSHTEIN, 2020))

O problema de inferência proposto pode ser representado pela Figura 10, em que $\boldsymbol{y}$ representa o sinal observado parametrizado por $\boldsymbol{\theta}$, os parâmetros desconhecidos do canal. A partir de $\boldsymbol{y}$ pretende-se estimar $\boldsymbol{x}$, os símbolos transmitidos parametrizados por $\boldsymbol{\Phi}$, que determina as probabilidades *a posteriori* dos símbolos. Tanto $\boldsymbol{\theta}$ como $\boldsymbol{\Phi}$ são parâmetros utilizados para maximizar a evidência, e portanto, ao ajustar $\boldsymbol{\Phi}$ determina-se a densidade a posteriori de $\boldsymbol{x}$. Ou seja, o modelo proposto funciona como um equalizador.

O modelo de sinal descrito na Seção 1 implica em que a função de massa de probabilidade (f.m.p.) de $\boldsymbol{x}$ seja dada por

$$p(\boldsymbol{x}) = 2^{-N}. \tag{3.1}$$

Definindo o conjunto de parâmetros $\boldsymbol{\theta} \triangleq \{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_R, \sigma_1^2, \ldots, \sigma_R^2\}$, a função densi-

dade de probabilidade (f.d.p.) condicional de $\boldsymbol{y}$ dados $\boldsymbol{x}$ e $\boldsymbol{\theta}$ pode ser escrita como

$$\begin{aligned} p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) &= \prod_{r=1}^{R} p_{\boldsymbol{\theta}}(\boldsymbol{y}_r|\boldsymbol{x}), \\ &= \prod_{r=1}^{R} \mathcal{N}\left(\boldsymbol{x} * \boldsymbol{h}_r,\ \sigma_r^2 \boldsymbol{I}_N\right), \\ &= \prod_{r=1}^{R} \frac{1}{(2\pi\sigma_r^2)^{\frac{N}{2}}} \cdot e^{-\frac{\|\boldsymbol{y}_r - \boldsymbol{x} * \boldsymbol{h}_r\|^2}{2\sigma_r^2}}. \end{aligned} \tag{3.2}$$

sendo a primeira igualdade decorrente da independência do ruído nos múltiplos subcanais, em que $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Xi})$ denota uma f.d.p. Gaussiana $N-$variada com vetor de médias $\boldsymbol{\mu}$ e matriz de covariância $\boldsymbol{\Xi}$, e $\boldsymbol{I}_N$ denota uma matriz identidade de tamanho N.

Para se determinar a estimativa de *máxima verossimilhança* (ML) de $\boldsymbol{\theta}$, é necessário calcular e maximizar

$$p_{\boldsymbol{\theta}}(\boldsymbol{y}) = \sum_{\boldsymbol{x} \in \{\boldsymbol{X}\}} p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) p(\boldsymbol{x}), \tag{3.3}$$

em que $\{\boldsymbol{X}\}$ denota o conjunto de todas as 2^N sequências possíveis $\boldsymbol{x}$, o que é inviável para $N \gg 1$.

O método de estimação variacional permite obter estimativas ML aproximadas substituindo $p_{\boldsymbol{\theta}}(\boldsymbol{y})$ por um limitante inferior chamado de limitante inferior variacional ou limitante inferior de evidência. Para isto, explora-se o fato, justificado a seguir, de que (KINGMA; WELLING, 2013):

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{y}) = \log \sum_{\boldsymbol{x}} p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) \tag{3.4}$$

$$= \log \sum_{\boldsymbol{x}} \left\{ p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) \frac{q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})}{q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})} \right\} \tag{3.5}$$

$$= \log \mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{x},\boldsymbol{y})} \left\{ \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})}{q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})} \right\} \tag{3.6}$$

$$\geq \mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{x},\boldsymbol{y})} \left\{ \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})}{q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})} \right\} \qquad \text{[Desigualdade de Jensen]} \tag{3.7}$$

$$= \mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})} \left\{ -\log q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y}) + \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) \right\} \tag{3.8}$$

$$\begin{aligned} &= -\mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})} \log \left[\frac{q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})}{p(\boldsymbol{x})} \right] - \mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})} \log p(\boldsymbol{x}) \\ &\quad + \mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}), \end{aligned} \tag{3.9}$$

$$= \underbrace{-D_{KL}\left[q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y}) || p(\boldsymbol{x})\right]}_{\triangleq A} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})}_{\triangleq B}, \tag{3.10}$$

$$\triangleq -\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y}). \tag{3.11}$$

em que $q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})$ é uma f.d.p. *arbitrária*, parametrizada por $\boldsymbol{\Phi}$, e $D_{KL}[\cdot||\cdot]$ indica a divergência de Kullback-Leibler (KULLBACK, 1997).

Em (3.4) expressou-se a evidência $p_{\boldsymbol{\theta}}(\boldsymbol{y})$ como a marginal da probabilidade conjunta $p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{y})$, que é determinada através de uma soma pois $\boldsymbol{x}$ é discreto. Em (3.5), por sua vez, multiplica-se e divide-se (3.4) por $q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})$, o que não altera a igualdade sob a hipótese de que esta f.m.p. não seja nula em nenhum ponto de seu suporte. Em seguida, em (3.6), o somatório de (3.5) é re-expresso como uma esperança. Isto, junto ao fato de a função logarítmica ser uma função côncava, possibilita a aplicação da Desigualdade de Jensen (vide Apêndice A.1) em (3.7). Finalmente, (3.8) resulta da aplicação da propriedade de divisão do logaritmos, e (3.9) da linearidade do operador esperança.

Desta forma, o método variacional, ao invés de maximizar diretamente $p_{\boldsymbol{\theta}}(\boldsymbol{y})$, minimiza a função custo $\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\Phi},\boldsymbol{y})$ conjuntamente em $\boldsymbol{\theta}$ e $\boldsymbol{\Phi}$. Pode-se mostrar (BLEI; KUCUKELBIR; MCAULIFFE, 2017) que, minimizando-se a mesma sobre $\boldsymbol{\theta}$ e todas as densidades condicionais possíveis $q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})$, obtém-se a estimativa ML de $\boldsymbol{\theta}$. Em outras palavras, como a evidência é maior ou igual ao seu limitante inferior, ao se determinar $\boldsymbol{\theta}$ que maximiza $-\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\Phi},\boldsymbol{y})$, a evidência é maximizada. A vantagem deste procedimento é que o cálculo de (3.11), diferentemente do de (3.3), não requer que se efetuem somas sobre todas as sequências $\boldsymbol{x}$ possíveis, o que tem complexidade computacional exponencial em N; este problema é substituído por um problema de maximizar parâmetros contínuos, que é resolvido por uma busca de gradiente e tem complexidade proporcional a N^2.

Tipicamente, ao se usar um VAE, tanto $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ como $q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})$ são aproximadas através de redes neurais. A densidade $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ é computada através de um *decodificador* e $q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})$ através de um *codificador*. Para o modelo de sinal em questão, não é necessária uma rede neural para implementar o decodificador, pois $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ possui expressão analítica. O codificador, por outro lado, é implementado em (CACIULARU; BURSHTEIN, 2020) através de uma rede neural que faz o papel de um equalizador.

Em suma, o equalizador VAE opera da seguinte forma (vide Algoritmo 1): as saídas do canal $\boldsymbol{y}$ são usadas para estimar os parâmetros $\boldsymbol{\theta}$ e $\boldsymbol{\Phi}$. Após isso, obtém-se uma estimativa da sequência de símbolos $\boldsymbol{x}$ usando a saída do codificador $q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})$.

Uma vez que os símbolos $x_j \in \{-1,1\}$, impõe-se que $q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})$ seja uma distribuição Bernoulli multivariada com independência estatística entre seus componentes:

$$q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y}) \triangleq \prod_{j=0}^{N-1} q_{\boldsymbol{\Phi}}(x_j|\boldsymbol{y}). \tag{3.12}$$

Denotando por $q_{\boldsymbol{\Phi},j}(\boldsymbol{y}) \triangleq q_{\boldsymbol{\Phi},j}(x_j=1|\boldsymbol{y})$ as probabilidades condicionais dos eventos $x_j = 1$ dado $\boldsymbol{y}$, tem-se que

$$q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y}) = \prod_{j=0}^{N-1} (q_{\boldsymbol{\Phi},j}(\boldsymbol{y}))^{(1+x_j)/2}(1-(q_{\boldsymbol{\Phi},j}(\boldsymbol{y}))^{(1-x_j)/2}. \tag{3.13}$$

A implementação do codificador é feita a partir de uma rede neural convolucional. A rede em questão possui uma camada de entrada convolucional, que aplica um filtro

FIR separadamente à cada entrada, soma as saídas desses filtros e aplica ao resultado a função de ativação *softsign*, definida por $f(x) = x/(|x|+1)$. A camada de saída determina $q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})$ aplicando a função de ativação *sigmoide* (que assegura que a saída esteja em $[0,\ 1]$) ao resultado da soma da saída da camada de entrada com uma combinação linear das entradas não processadas. Verificou-se experimentalmente que o uso da função de ativação *softsign* na camada de entrada leva a melhores resultados que o uso de funções como *L*eakyReLU e *t*anh.

3.0.1 Cálculo da Função Objetivo

Para se deduzir a expressão de $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y})$ (Eq. 3.11), observa-se inicialmente que

$$\begin{aligned} A &\triangleq \sum_{x} q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y}) \cdot (\log p(\boldsymbol{x}) - \log q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})) \\ &= \sum_{x} q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y}) \cdot (-N\log 2 - \log q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})) \\ &= -N\log 2 + \mathcal{H}[q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})], \end{aligned} \tag{3.14}$$

em que $\mathcal{H}[q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})]$ denota a entropia de $q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})$ dada por

$$\begin{aligned} \mathcal{H}[q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})] = \mathcal{H}\left[\prod_{j=0}^{N-1} q_{\boldsymbol{\Phi}}(x_j|\boldsymbol{y})\right] = \sum_{j=0}^{N-1} \mathcal{H}\left[q_{\boldsymbol{\Phi}}(x_j|\boldsymbol{y})\right] \\ = -\sum_{j=0}^{N-1} \{q_{\boldsymbol{\Phi},j}(\boldsymbol{y})\log q_{\boldsymbol{\Phi},j}(\boldsymbol{y}) + \\ (1 - q_{\boldsymbol{\Phi},j}(\boldsymbol{y}))\log(1 - q_{\boldsymbol{\Phi},j}(\boldsymbol{y}))\}\,. \end{aligned} \tag{3.15}$$

Para o termo B, tem-se:

$$\begin{aligned} B &= \mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})}\left[-\sum_{r=1}^{R}\left(\frac{N}{2}\log\left(2\pi\sigma_r^2\right) + \frac{\|\boldsymbol{y}_r - \boldsymbol{x} * \boldsymbol{h}_r\|^2}{2\sigma_r^2}\right)\right] \\ &= -\frac{N}{2}\log(2\pi) - \frac{N}{2}\sum_{r=1}^{R}\log(\sigma_r^2) - \sum_{r=1}^{R}\frac{1}{2\sigma_r^2}\underbrace{\mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})}\left[\|\boldsymbol{y}_r - \boldsymbol{x} * \boldsymbol{h}_r\|^2\right]}_{C_r}. \end{aligned} \tag{3.16}$$

Os termos C_r, interpretáveis como a esperança do erro quadrático de estimação do sinal recebido por cada subcanal, podem ser calculados analiticamente. Manipulando-se a expressão de C_r, pode-se escrever

$$\begin{aligned} C_r = \sum_{n=0}^{N-1}\Bigg\{ y_{n,r}^2 - 2\left(y_{n,r}\sum_{k=0}^{L-1}\mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})}[x_k]\cdot h_{n-k,r}\right) + \\ \sum_{k,l=0}^{N-1}\mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})}[x_k x_l]\cdot h_{n-k,r}h_{n-l,r}\Bigg\}, \end{aligned} \tag{3.17}$$

em que $h_{n,r} = 0$ para $n \geq L$.

Calculando-se as esperanças em (3.17), obtém-se que

$$\mathbb{E}_{q_{\boldsymbol{\Phi}(\boldsymbol{x}|\boldsymbol{y})}}[x_k] = 2q_{\boldsymbol{\Phi},k}(\boldsymbol{y}) - 1, \tag{3.18}$$

$$\mathbb{E}_{q_{\boldsymbol{\Phi}(\boldsymbol{x}|\boldsymbol{y})}}[x_k^2] = 1, \tag{3.19}$$

e, para $k \neq l$,

$$\begin{aligned}\mathbb{E}_{q_{\boldsymbol{\Phi}(\boldsymbol{x}|\boldsymbol{y})}}[x_k x_l] &= \mathbb{E}_{q_{\boldsymbol{\Phi}(\boldsymbol{x}|\boldsymbol{y})}}[x_k] \cdot \mathbb{E}_{q_{\boldsymbol{\Phi}(\boldsymbol{x}|\boldsymbol{y})}}[x_l] \\ &= \left(2q_{\boldsymbol{\Phi},k}(\boldsymbol{y}) - 1\right)\left(2q_{\boldsymbol{\Phi},l}(\boldsymbol{y}) - 1\right). \end{aligned} \tag{3.20}$$

Substituindo-se (3.18), (3.19) e (3.20) em (3.17) é possível obter uma expressão explícita para C_r. Para isto, observe que (CACIULARU; BURSHTEIN, 2020):

$$\begin{aligned}\sum_{k,l=0}^{N-1} \mathbb{E}_{q_{\boldsymbol{\Phi}(\boldsymbol{x}|\boldsymbol{y})}}[x_k x_l] \cdot h_{n-k,r} h_{n-l,r} = \\ \left\{\sum_{k=0}^{N-1} \mathbb{E}_{q_{\boldsymbol{\Phi}(\boldsymbol{x}|\boldsymbol{y})}}[x_k] \cdot h_{n-k,r}\right\}^2 + \\ \sum_{k=0}^{N-1} h_{n-k,r}^2 \left\{1 - \mathbb{E}^2_{q_{\boldsymbol{\Phi}(\boldsymbol{x}|\boldsymbol{y})}}[x_k]\right\}. \end{aligned} \tag{3.21}$$

Consequentemente,

$$C_r = \sum_{k=0}^{N-1} \|\boldsymbol{y}_{n,r}\|^2 - 2\alpha_{n,r} + \beta_{n,r}, \tag{3.22}$$

em que

$$\alpha_{n,r} = \sum_{k=0}^{N-1} y_{n,r} h_{n-k} \cdot (2q_{\boldsymbol{\Phi},k}(\boldsymbol{y}) - 1), \tag{3.23}$$

e

$$\beta_{n,r} = \sum_{k=0}^{N-1} h_{n-k,r} \cdot (2q_{\boldsymbol{\Phi},k}(\boldsymbol{y}) - 1) + \tag{3.24}$$

$$\sum_{k=0}^{N-1} (h_{n-k,r})^2 \left[4q_{\boldsymbol{\Phi},k}(\boldsymbol{y}) - 4q^2_{\boldsymbol{\Phi},k}(\boldsymbol{y})\right]. \tag{3.25}$$

Vale ressaltar que os termos C_r são usados para determinar $h_{n,r}$ durante a adaptação do algoritmo, e não o oposto. Ou seja, $h_{n,r}$ é determinado minimizando-se C_r.

3.0.2 Treinamento

O treinamento do VAE envolve minimizar $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y}) = -A - B$ em relação a $\boldsymbol{\theta} = \{\boldsymbol{h}, \sigma_1^2, \ldots, \sigma_R^2\}$ e a $\boldsymbol{\Phi}$. Para se calcular o mínimo em relação às variâncias σ_r^2, $r = 1, \ldots, R$, que só afetam o termo B, calcula-se o gradiente deste termo e iguala-se o

resultado a zero. Disto se obtém que o valor ótimo de σ_r^2 é dado por C_r/N. Utilizando este resultado e (3.14), pode-se escrever que

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y}) = \frac{N}{2} \sum_{r=1}^{R} \{\log(C_r)\} - \mathcal{H}[q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})] + c, \tag{3.26}$$

em que $\mathcal{H}[q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})]$ é dado em (3.15), C_r em (3.22), e c é uma constante que não depende dos parâmetros livres $(\boldsymbol{\theta}, \boldsymbol{\Phi})$.

Os parâmetros desconhecidos remanescentes ($\boldsymbol{h}$ e $\boldsymbol{\Phi}$) são estimados através da minimização baseada em gradiente descendente de (3.26). Cabe notar que não é necessário calcular analiticamente os gradientes; nas simulações relatadas a seguir, os mesmos são calculados numericamente pelo software *Tensorflow* (ABADI et al., 2016).

Finalmente, note que não é necessário se conhecer o valor exato da ordem do canal $\boldsymbol{L}$ para se executar o algoritmo; ao invés disso, basta se ter uma limite superior para o qual a resposta ao impulso do canal verdadeiro possa ser bem aproximada.

Estimação dos Símbolos: uma vez determinado $\boldsymbol{\Phi}$ que minimiza (3.26), a sequência de símbolos $\boldsymbol{x}$ pode então ser estimada a partir de $q_{\boldsymbol{\Phi},j}(\boldsymbol{y})$ (Eq. 3.12), o que pode ser interpretado como uma decodificação suave (*soft decoding*) (PROAKIS; SALEHI, 2008) desta sequência.

O Algoritmo 1 sumariza o método descrito nesta seção.

Algoritmo 1 Algoritmo VAE

Dados $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_R$ e t_{max} (número de épocas):
for $t \leftarrow 1$ to t_{max} **do**
- Calcule $q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})$ através do codificador (Fig. 11).
- Calcule C_r , $1 \leq r \leq R$ usando a Eq. 3.22.
- Determine numericamente os parâmetros $\boldsymbol{\theta}$ (codificador) e $\boldsymbol{\Phi}$ (decodificador) que minimizam a função custo $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y})$ (Eq. 3.26).

end for
Determine uma estimativa de $\boldsymbol{x}$ a partir da f.m.p. $q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})$.

3.0.3 Implementação Alternativa do Algoritmo VAE SIMO

Uma implementação alternativa do algoritmo VAE (vide Algoritmo 2) consiste em substituir a esperança dos termos C_r em (3.16) por uma aproximação *estocástica*, i.e., fazendo

$$C_r \approx \|\boldsymbol{y}_r - \tilde{\boldsymbol{x}} * \boldsymbol{h}_r\|^2, \tag{3.27}$$

em que $\tilde{\boldsymbol{x}}$ é uma amostra de $q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})$, que é determinada pelo codificador.

Esse método tem menor complexidade computacional, pois elimina a necessidade de se calcularem as múltiplas somas de convolução de (3.17).

Algoritmo 2 Implementação alternativa do algoritmo VAE

Dados $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_R$ e t_{max} (número de épocas):
for $t \leftarrow 1$ to t_{max} **do**

- Calcule $q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})$ através do codificador (Fig. 11).
- Gere uma amostra $\tilde{\boldsymbol{x}}$ de $q_{\boldsymbol{\Phi}}(\boldsymbol{x}|\boldsymbol{y})$.
- Usando $\tilde{\boldsymbol{x}}$ e a expressão para C_r , $1 \leq r \leq R$ da Eq. 3.27, minimize numericamente a função custo $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{y})$ (Eq. 3.26).

end for
Retorne $\tilde{\boldsymbol{x}}$ (estimativa dos símbolos).

4 Resultados Experimentais

Num primeiro experimento, para avaliar o desempenho dos algoritmos descritos nas Seções 3 e 3.0.3, realizaram-se simulações numéricas que empregaram 600 realizações independentes, nas quais geraram-se dados sintéticos de acordo com o modelo dado pela Equação 1.7. Os programas foram implementados na linguagem Python e utilizaram os pacotes NumPy e Tensorflow.

Utilizou-se um canal SIMO com $R = 2$ saídas; os subcanais empregados têm resposta ao impulso

$$\boldsymbol{h}_1 = [0.0661,\ 0.34376866,\ -0.9318,\ -0.0778,\ 0.0566],$$
$$\boldsymbol{h}_2 = [0.1594,\ -0.3816,\ 0.8888,\ -0.1836,\ -0.0725].$$

Os algoritmos empregaram 128 amostras de treinamento. Observe que, por se tratarem de algoritmos *cegos*, o treinamento utiliza apenas amostras dos sinais recebidos. As camadas convolucionais dos algoritmos empregaram filtros com 5 coeficientes. O treinamento dos algoritmos usou o otimizador ADAM (KINGMA; BA, 2014), com taxa de aprendizado 0,05 durante 300 *épocas*, utilizando sempre os mesmos dados para treinamento. Para o teste dos algoritmos, foram utilizadas sequências distintas das de treinamento com 1.000 amostras de sinal recebido.

O *codificador* VAE utilizou a estrutura mostrada na Figura 11. Vale ressaltar que, para canais SIMO, a estrutura com duas camadas convolucionais proposta em (CACIULARU; BURSHTEIN, 2020) exibiu desempenho semelhante à utilizada.

Para comparação de desempenho, simularam-se os resultados, sob as mesmas condições, dos algoritmos:
i) NNCMA (DAI, 2001), sem linearização, ou seja, com o parâmetro $K = 1$;
ii) KLD, um novo algoritmo que utiliza como função custo apenas o termo A de (3.11);
iii) LMMSE, definido com o Filtro de Wiener (DINIZ, 2012) com 5 coeficientes, determinado através dos parâmetros exatos ($\boldsymbol{h}_1$, $\boldsymbol{h}_2$, σ_1^2, σ_2^2).

Os valores de σ_r^2 foram supostos iguais para todos os subcanais e foram determinados de modo a produzir a relação sinal-ruído (SNR) desejada.

A Figura 12 mostra o desempenho dos algoritmos supracitados em termos da taxa média de erro de bit (BER) em função da SNR. Como se pode observar, os algoritmos VAE têm um desempenho bastante superior ao do NNCMA, cujo desempenho converge para um patamar com BER da ordem de $3 \cdot 10^{-2}$ para SNRs superiores a 10dB, resultado semelhante ao reportado em (DAI, 2001). No entanto, os algoritmos VAE exibem uma

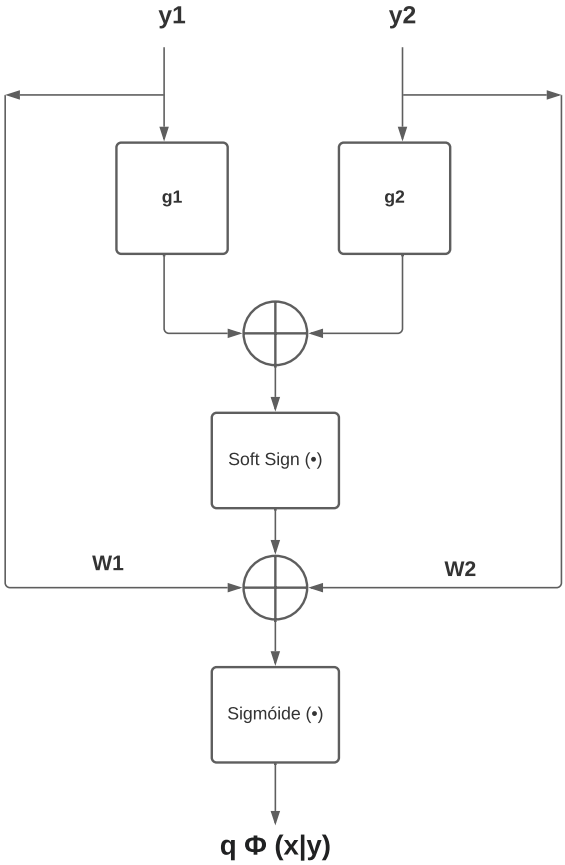

Figura 11 – Estrutura do *codificador*, comum para os algoritmos das Seções 3 e 3.0.3. Cada entrada é processada por um filtro separado (g_1 e g_2). W_1 e W_2 são pesos adaptáveis.

perda de desempenho de cerca de 3dB em relação ao algoritmo LMMSE (para BER de 10^{-4}).

Os algoritmos VAE das Seções 3 e 3.0.3, surpreendentemente, exibiram desempenhos semelhantes, apesar da aproximação empregada pelo algoritmo da Seção 3.0.3. Por fim, observe que o algoritmo KLD, que não leva em conta o erro de reconstrução produzido pelo decodificador, tem um desempenho semelhante ao dos algoritmos VAE.

Num segundo experimento, avaliou-se o desempenho dos algoritmos de equalização lineares para o mesmo canal através de simulações numéricas com 300 realizações independentes. Todos os equalizadores utilizaram 6 *taps*. Os algoritmos treinados (RLS e LMS) usaram 300 amostras de treinamento, atraso de 3 amostras e constantes de adaptação $\lambda = 0,99$ e $\mu = 10^{-2}$, respectivamente. Foram testadas duas versões do CMA: a clássica (baseada no método gradiente descendente), com constante de adaptação $\nu = 10^{-3}$, valor máximo que não provocou a divergência do algoritmo, e a baseada no otimizador ADAM, com os parâmetros descritos na Seção 1.1.3. O número de amostras usadas no treinamento foi de 300, 3.000 e 30.000. Para a estimação da BER, os equalizadores foram mantidos constantes e foi transmitido um total de 1.000 símbolos pelo canal.

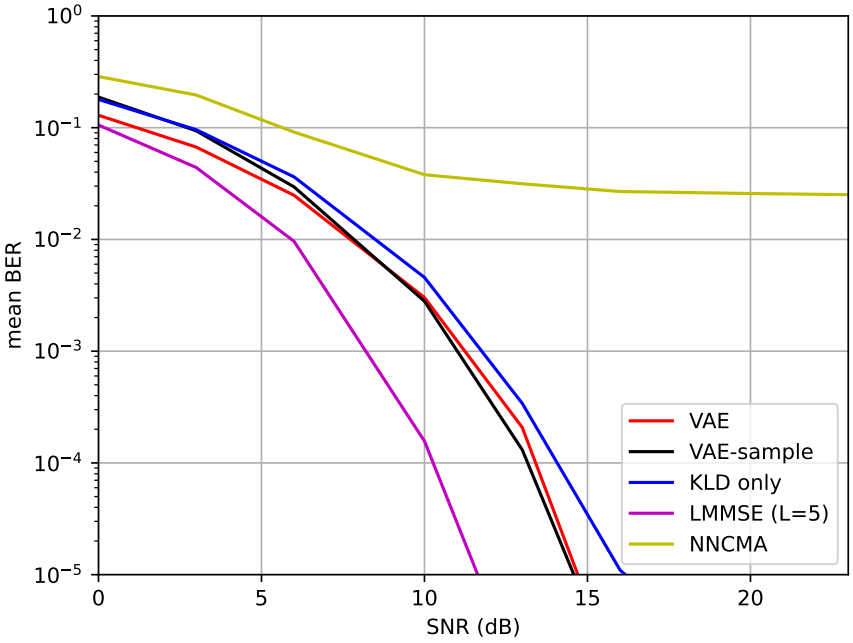

Figura 12 – Taxa média de erro de bit (BER) ao longo de 600 realizações independentes para os algoritmos VAE (Algoritmo 1), VAE-sample (Algoritmo 2) e outros descritos na Seção V.

Como se pode observar na Figura 13, o CMA-ADAM obteve um desempenho muito superior CMA clássico ao se utilizar o mesmo número de amostras de treinamento. Não foi possível simular o CMA-ADAM com 30.000 amostras de treinamento mas, neste caso, o CMA tradicional já é quase equivalente ao RLS/LMS que, por sua vez, aproximam a solução ótima de mínimos quadrados.

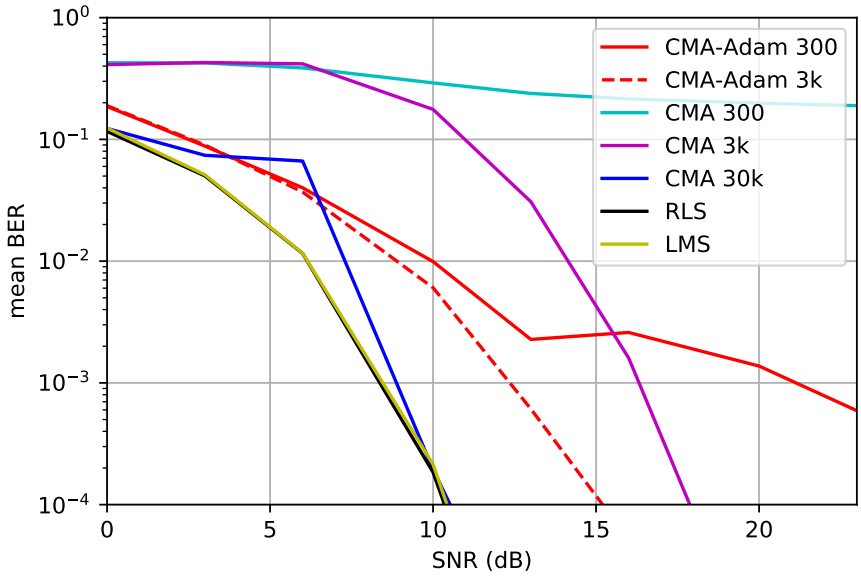

Figura 13 – Taxa média de erro de bit (BER) ao longo de 300 realizações independentes para o CMA, CMA Adam, e os algoritmos RLS e LMS. Os números após os rótulos com os nomes dos algoritmos indicam os números de símbolos usados na adaptação dos mesmos. Para os algoritmos LMS e RLS foram usados 300 símbolos.

5 Conclusões

Propôs-se neste estudo uma extensão do algoritmo de equalização cega baseado em autocodificadores variacionais de (CACIULARU; BURSHTEIN, 2020) para canais SIMO e duas modificações deste algoritmo: uma nova versão que emprega aproximações amostrais de certas esperanças e outra com um função custo simplificada. Os três algoritmos propostos exibem desempenhos semelhantes, e bastante superiores aos do método NNCMA, sendo, porém, inferiores em cerca de 3dB em relação ao filtro de Wiener, calculado com os parâmetros exatos do canal.

Propôs-se ainda uma nova versão do CMA que utiliza o otimizador ADAM ao invés do método clássico baseado em gradiente estocástico descendente. Como se pôde observar, o uso do otimizado ADAM leva a uma grande melhoria de desempenho, que pode ser creditada a uma maior taxa de convergência do algoritmo.

Como sugestões para continuação deste trabalho, propõe-se:

1. Análise do desempenho de estruturas alternativas para o codificador VAE, uma vez que, surpreendentemente, o uso de mais de uma camada convolucional não provocou melhoria de desempenho para o caso SIMO.

2. Extensão do algoritmo VAE para sistemas MIMO.

Referências

ABADI, M. et al. Tensorflow: A system for large-scale machine learning. In: *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. [S.l.: s.n.], 2016. p. 265–283. Citado na página 24.

BALESTRIERO, R.; PARIS, S.; BARANIUK, R. Analytical probability distributions and exact expectation-maximization for deep generative networks. *Advances in neural information processing systems*, v. 33, p. 14938–14949, 2020. Citado na página 17.

BENVENISTE, A.; GOURSAT, M. Blind equalizers. *IEEE Transactions on communications*, IEEE, v. 32, n. 8, p. 871–883, 1984. Citado na página 2.

BLEI, D. M.; KUCUKELBIR, A.; MCAULIFFE, J. D. Variational inference: A review for statisticians. *Journal of the American statistical Association*, Taylor & Francis, v. 112, n. 518, p. 859–877, 2017. Citado na página 21.

CACIULARU, A.; BURSHTEIN, D. Blind channel equalization using variational autoencoders. In: IEEE. *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*. [S.l.], 2018. p. 1–6. Citado na página 3.

CACIULARU, A.; BURSHTEIN, D. Unsupervised linear and nonlinear channel equalization and decoding using variational autoencoders. *IEEE Transactions on Cognitive Communications and Networking*, IEEE, v. 6, n. 3, p. 1003–1018, 2020. Citado 7 vezes nas páginas 3, 17, 19, 21, 23, 27 e 31.

DAI, X. H. Cma-based nonlinear blind equaliser modelled by a two-layer feedforward neural network. *IEE Proceedings-Communications*, IET, v. 148, n. 4, p. 243–248, 2001. Citado 2 vezes nas páginas 3 e 27.

DING, Z.; LI, Y. *Blind equalization and identification*. [S.l.]: CRC press, 2018. Citado na página 3.

DINIZ, P. S. R. *Adaptive Filtering: Algorithms and Practical Implementation*. [S.l.]: Springer US, 2012. (SpringerLink : Bücher). Citado na página 27.

DOERSCH, C. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016. Citado na página 2.

DRAGOMIR, S.; PEČARIĆ, J.; SÁNDOR, J. A note on the jensen-hadamard inequality. *Mathematica-Revue d'analyse numérique et de théorie de l'approximation. L'analyse numérique et la théorie de l'approximation*, v. 19, n. 1, p. 29–34, 1990. Citado na página 39.

EZHILARASAN, E.; DINAKARAN, M. A review on mobile technologies: 3g, 4g and 5g. In: IEEE. *2017 second international conference on recent trends and challenges in computational models (ICRTCCM)*. [S.l.], 2017. p. 369–373. Citado na página 1.

FORNEY, G. D. The viterbi algorithm. *Proceedings of the IEEE*, Ieee, v. 61, n. 3, p. 268–278, 1973. Citado na página 2.

GHOSH, M.; WEBER, C. L. Maximum-likelihood blind equalization. *Optical Engineering*, International Society for Optics and Photonics, v. 31, n. 6, p. 1224–1228, 1992. Citado na página 2.

GODARD, D. Self-recovering equalization and carrier tracking in two-dimensional data communication systems. *IEEE Transactions on Communications*, v. 28, n. 11, p. 1867–1875, 1980. Citado na página 2.

GODARD, D. Self-recovering equalization and carrier tracking in two-dimensional data communication systems. *IEEE Transactions on Communications*, v. 28, n. 11, p. 1867–1875, 1980. Citado na página 3.

HANNA, S.; DICK, C.; CABRIC, D. *Signal Processing Based Deep Learning for Blind Symbol Decoding and Modulation Classification*. 2021. Disponível em: <https://arxiv.org/abs/2106.10543>. Citado na página 2.

HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Bookman Editora, 2001. Citado na página 13.

JOHNSON, R. et al. Blind equalization using the constant modulus criterion: A review. *Proceedings of the IEEE*, IEEE, v. 86, n. 10, p. 1927–1950, 1998. Citado na página 2.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Citado 3 vezes nas páginas 3, 9 e 27.

KINGMA, D. P.; WELLING, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013. Citado 3 vezes nas páginas 2, 17 e 20.

KINGMA, D. P.; WELLING, M. et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, Now Publishers, Inc., v. 12, n. 4, p. 307–392, 2019. Citado 3 vezes nas páginas 15, 16 e 17.

KULLBACK, S. *Information Theory and Statistics*. [S.l.]: Dover Publications, 1997. (A Wiley publication in mathematical statistics). Citado na página 20.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015. Citado 2 vezes nas páginas 2 e 14.

LITTLE, M. A. *Machine Learning for Signal Processing: Data Science, Algorithms, and Computational Statistics*. [S.l.]: Oxford University Press, 2019. Citado na página 2.

NACHMANI, E.; BE'ERY, Y.; BURSHTEIN, D. Learning to decode linear codes using deep learning. In: IEEE. *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. [S.l.], 2016. p. 341–346. Citado na página 3.

NG, A. et al. Sparse autoencoder. *CS294A Lecture notes*, v. 72, n. 2011, p. 1–19, 2011. Citado na página 17.

O'SHEA, T.; HOYDIS, J. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, IEEE, v. 3, n. 4, p. 563–575, 2017. Citado na página 3.

PINAYA, W. H. L. et al. Autoencoders. In: *Machine learning*. [S.l.]: Elsevier, 2020. p. 193–208. Citado na página 15.

PROAKIS, J. G.; SALEHI, M. *Digital Communications*. [S.l.]: McGraw-Hill, 2008. (McGraw-Hill International Edition). Citado 4 vezes nas páginas 1, 6, 9 e 24.

RAHMATALLAH, Y.; MOHAN, S. Peak-to-average power ratio reduction in ofdm systems: A survey and taxonomy. *IEEE communications surveys & tutorials*, IEEE, v. 15, n. 4, p. 1567–1592, 2013. Citado na página 1.

ROMANO, J. et al. *Unsupervised Signal Processing: Channel Equalization and Source Separation*. [S.l.]: CRC Press, 2018. ISBN 9781420019469. Citado na página 2.

SAMUEL, N.; DISKIN, T.; WIESEL, A. Deep mimo detection. In: IEEE. *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. [S.l.], 2017. p. 1–5. Citado na página 3.

SAYED, A. H. *Adaptive filters*. [S.l.]: John Wiley & Sons, 2011. Citado 3 vezes nas páginas 7, 8 e 9.

SHALVI, O.; WEINSTEIN, E. Super-exponential methods for blind deconvolution. *IEEE Transactions on Information Theory*, IEEE, v. 39, n. 2, p. 504–519, 1993. Citado na página 2.

SHLEZINGER, N. et al. Viterbinet: A deep learning based viterbi algorithm for symbol detection. *IEEE Transactions on Wireless Communications*, IEEE, v. 19, n. 5, p. 3319–3331, 2020. Citado na página 2.

TSOUKALAS, L. H.; UHRIG, R. E. *Fuzzy and neural approaches in engineering*. [S.l.]: John Wiley & Sons, Inc., 1996. Citado na página 11.

VINCENT, P. et al. Extracting and composing robust features with denoising autoencoders. In: *Proceedings of the 25th international conference on Machine learning*. [S.l.: s.n.], 2008. p. 1096–1103. Citado na página 17.

WANG, X.; CHEN, R. Blind turbo equalization in gaussian and impulsive noise. *IEEE Transactions on Vehicular Technology*, IEEE, v. 50, n. 4, p. 1092–1105, 2001. Citado na página 2.

WU, Y. et al. Comparison of terrestrial dtv transmission systems: the atsc 8-vsb, the dvb-t cofdm, and the isdb-t bst-ofdm. *IEEE Transactions on Broadcasting*, IEEE, v. 46, n. 2, p. 101–113, 2000. Citado na página 1.

YOU, C.; HONG, D. Nonlinear blind equalization schemes using complex-valued multilayer feedforward neural networks. *IEEE transactions on neural networks*, IEEE, v. 9, n. 6, p. 1442–1455, 1998. Citado na página 2.

Apêndices

APÊNDICE A – Demonstrações

A.1 Desigualdade de Jensen

Teorema (Desigualdade de Jensen) ***(DRAGOMIR; PEČARIĆ; SÁNDOR, 1990):*** *seja f:(a,b)* $\to \mathbb{R}$ *duas vezes diferenciável. Se* $f''(x) \geq 0$ *(função convexa) em todo intervalo (a,b), então para quaisquer* $x_1, x_2, ..., x_n \in (a, b)$ *vale:*

$$\frac{f(x_1) + f(x_2) + ... + f(x_n)}{n} \geq f\left(\frac{x_1 + x_2 + ... + x_n}{n}\right). \tag{A.1}$$

Demonstração: A prova será por indução sobre n. O caso $n = 1$ não oferece resistência. Suponha então que a desigualdade valha para quaisquer $n - 1$ reais no intervalo (a, b). Inicialmente, fixa-se $x_1, x_2, ..., x_{n-1}$ e chama-se $x_n = x$. Agora, se faz $x_1 + x_2 + ... + x_{n-1} = l$ e $f(x_1) + f(x_2) + ... + f(x_n - 1) = k$. É necessário provar que:

$$\frac{k + f(x)}{n} \geq f\left(\frac{l + x}{n}\right), \quad \forall x \in (a, b). \tag{A.2}$$

Defina então a função:

$$g(x) = \frac{k + f(x)}{n} - f\left(\frac{l + x}{n}\right). \tag{A.3}$$

Derivando, obtém-se:

$$g'(x) = \frac{f'(x)}{n} - \frac{1}{n} f'\left(\frac{l + x}{n}\right). \tag{A.4}$$

Se $x = \dfrac{l + x}{n} \Rightarrow x = \dfrac{l}{n - 1}$, tem-se que $g'(x) = 0$. Utilizando agora o fato de que $f'(x)$ é não decrescente em (a,b) pois $f''(x) \geq 0$) pode-se inferir que, se $x < \frac{l}{n-1}$, vale que $g'(x) \leq 0$. Por outro lado, se $x > \frac{l}{n-1}$, segue que $g'(x) \geq 0$.

Através do cálculo, pode-se concluir que $x = \frac{l}{n-1}$ é um ponto de mínimo global para g(x) no intervalo (a,b). Assim, tem-se que

$$g(x) \geq g\left(\frac{l}{n - 1}\right) = \frac{k}{n} - \frac{(n - 1) f\left(\frac{l}{n-1}\right)}{n} \geq 0, \tag{A.5}$$

pois

$$\frac{k}{n - 1} \geq f\left(\frac{l}{n - 1}\right) \tag{A.6}$$

pela hipótese de indução.

As condições para a igualdade dependem da função $f(x)$. No caso mais comum em que $f''(x) > 0$ no intervalo (a, b) pode-se, pela demostração acima, concluir que a igualdade só ocorre se $x_1 = x_2 = ... = x_n$.