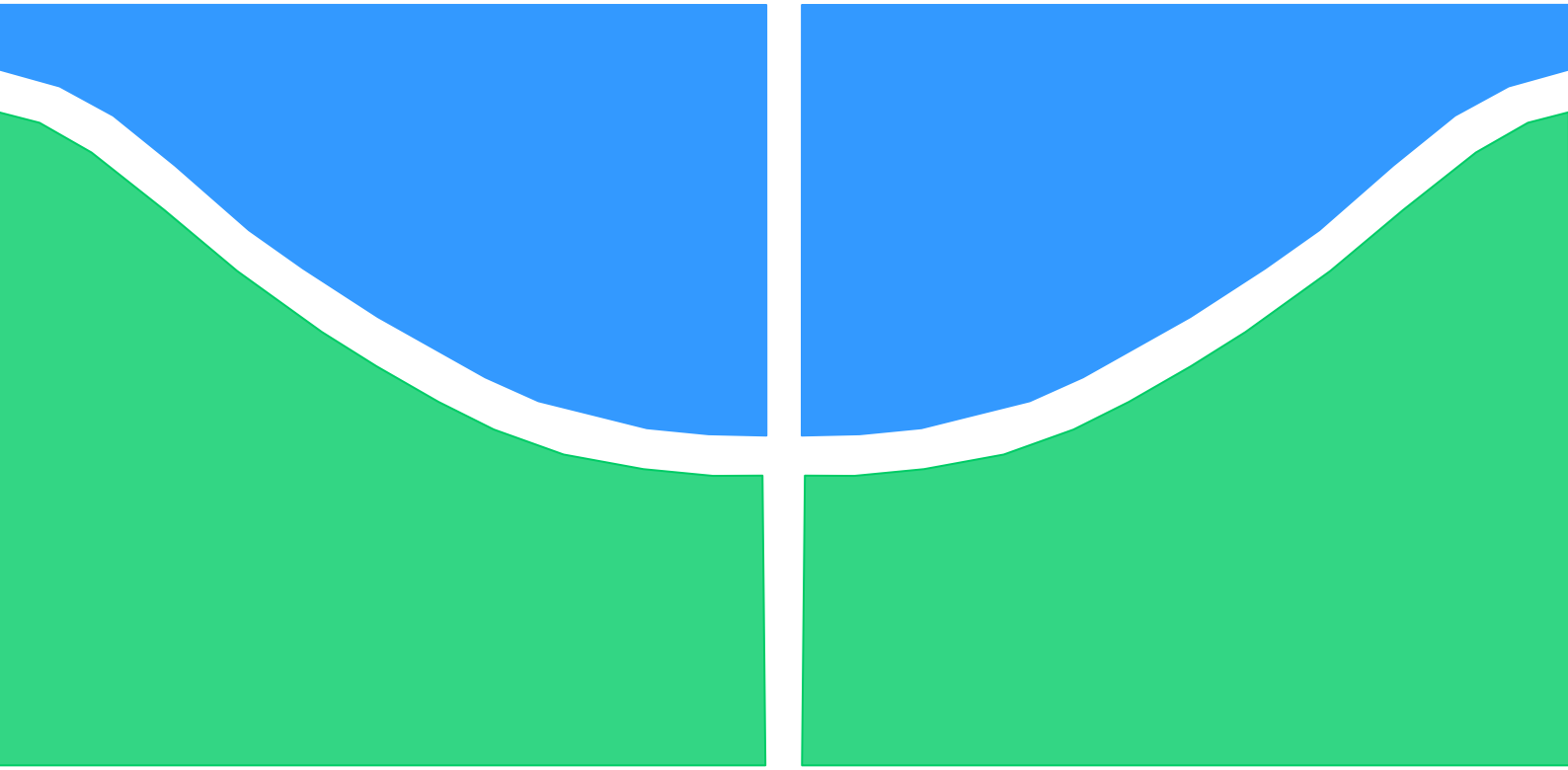

TRABALHO DE GRADUAÇÃO

ARQUITETURA DE REDE NEURAL DE BASE RADIAL APLICADA AO PROBLEMA DE EQUALIZAÇÃO CEGA

JEAN MAKITA KIBALA

Brasília, julho de 2017

UNIVERSIDADE DE BRASILIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

TRABALHO DE GRADUAÇÃO

ARQUITETURA DE REDE NEURAL DE BASE RADIAL APLICADA AO PROBLEMA DE EQUALIZAÇÃO CEGA

JEAN MAKITA KIBALA

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de rede de comunicação

Banca Examinadora

Prof. JOÃO PAULO LEITE, UnB/ Eng. Elétrica (Orientador) _____

Prof. DANIEL GUERREIRO E SILVA, UnB/ Eng. Elétrica _____

Prof. LEONALDO RAX, UnB/ Eng. Elétrica _____

Citação

« 32. Hoje nós enviamos nossos filhos à escola, para terem entendimento. Então, depois que eles concluem o curso de primeiro grau, nós os enviamos ao curso de segundo grau, para um melhor entendimento de conhecimento. Então, depois que eles o terminam, algumas crianças são mesmo suficientemente afortunadas para ir à universidade, e passam através da universidade, para completar sua educação e seu entendimento de conhecimento. O que eles pretendem… Muitas vezes, para obter um emprego, você tem que ter no mínimo um entendimento de nível de segundo grau, ou uma educação universitária, ou assim por diante.

33. O sábio Salomão ainda nos disse que: "não te estribes nisto, não no nosso próprio entendimento; não aprender destas coisas". Porque, nós nos perguntamos o motivo pelo qual ele nos diria tal coisa como essa. É porque nosso moderno entendimento é, usualmente, a sabedoria do homem, a qual é contrária à Palavra de Deus. E eu penso que é isto que Salomão estava tentando aconselhar a seus filhos. Não era para ser iletrados, mas não se estribar em seu próprio entendimento.

34. E eu penso que esta seria uma boa exortação hoje, se nós disséssemos a nossos filhos, e aos filhos de Deus, que, está tudo bem ter uma educação, não há nada contra isto; mas quando essa educação é contrária à Palavra de Deus, então estribe-se na Palavra e deixe sua educação ir-se, está vendo, por causa da Palavra. A educação sustentará e lhe dará um bom emprego, provavelmente lhe dará uma boa posição entre pessoas intelectuais, mas isso está bem, o que provavelmente será uma grande ajuda para você, ajudá-lo-á em suas finanças e sua – sua subsistência, talvez fará a vida um pouco melhor para você.

35. Mas lembre-se de uma coisa, meu filho, você tem que morrer. Não importa quanta educação você tenha, quanta cultura você é capaz de acumular, você ainda tem que encarar a morte, porque está escrito que: "O homem tem que morrer, e depois disso, vem o Julgamento". E Deus, quando…A morte não é tão má, mas vir ao Julgamento, é a parte ruim. Agora, você pode morrer, "Depois disso, porém, é o Julgamento". E Deus não lhe irá perguntar quanta instrução você tinha quando estava aqui na terra, quanto conhecimento você acumulou, quer você tenha seu título de Bacharel de Artes, ou qualquer que seja o diploma que você possa ter tido, mesmo como um ministro. Isto não vai ser requerido de você.

36. Será, porém, requerido de você, o que você fez acerca do entendimento da Palavra de Deus. Aqui é onde a exigência vem, por causa d'Isto. Sua educação é ótima, mas a Palavra de Deus é Vida. "Minha Palavra é Vida", e sabe-La é Vida. E Ele, Ele disse: "Conhecei-O". Ele é a Palavra. Assim você só poderá conhece-lo pela Palavra, porque Ele é a Palavra. E esta é a única maneira como você O conhecerá: através de Sua Palavra. »

William Marriom Branham

Dedicatória:

Dedico este trabalho para minha mãe, Elysée Miandabu Buloba, uma mulher negra, mulher africana da República Democrática do Congo. Uma mulher trabalhadeira e também uma grande fonte de inspiração. Quando era menino, sem nenhuma ideia do mundo real, sempre me falava o valor dos estudos; lembro-me das suas palavras e ainda posso ouvi-la dizer: " Étudie Jean, parce que un jour, tu seras un pére de famille ", pode ser traduzido: "Estude Jean, porque um dia, você será um pai de família". Então, para todos os leitores, saibam que este é o fruto de um investimento profundo de ensinamento de uma mãe para seu filho amado.

Jean Makita Kibala

Agradecimentos:

Agradeço em primeiro lugar a Deus por ter me dado o folego da vida, senão, não estaria aqui. E em segundo, a meu pai Josué Kibala Kisenge por ter me trazido no mundo. Á meus irmãos, Lievin Buloba, Justin Buloba e Frederick Kibala, e irmãs, Sarah Kibala e Hattie Kibala, que me apoiaram durante todo tempo dos meus estudos. Agradeço à minha esposa Sulany Makita e à minha sogra Célia Moura, que fizeram dos meus estudos uma prioridade, e aos meus amigos Christel Bulembi, Joseph Tshivuila, Michée Katuku, Vinicius Oliveira da Silva, Jonathan Mpoto e Josué Kabongo pelo apoio moral (bana mayi). Sou grato ao governo Brasileiro pela oportunidade de me tornar um aluno internacional, e ao governo da República Democrática do Congo pela confiança. À Igreja A voz de Deus, particularmente ao pastor João Pereira dos Santos, pelo companheirismo e acolhimento. Agradeço muito a todos meus professores da graduação, especialmente ao meu professor e orientador João Paulo Leite, uma pessoa diferenciada, aprendi muito com ele, que me dizia: "só em bater olho nada vai sair, precisa sentar, pegar papel, caneta e começar a escrever cada equação", com essas palavras que consegui fazer esse trabalho. Também agradeço à minha cunhada Suzany Moura por aceitar acompanhar a redação. Por fim, agradeço a todos aqueles que direto ou indiretamente contribuíram para o desenvolvimento deste trabalho.

Jean Makita Kibala

RESUMO

Neste trabalho mostramos o funcionamento da equalização cega usando a rede neural artificial, função de base radial (RBF). Explorando as características do sinal de entrada, a rede neural é utilizada para prever qual a saída esperada do filtro inverso (filtro de equalização). A obtenção dos coeficientes do equalizador é realizada utilizando-se os algoritmos *least squares* (LS) e *improved least squares* (ILS), sendo o último de desempenho superior ao primeiro em ambientes com ruído do tipo branco. Os resultados de simulação mostrados incluem a capacidade da RBF em predizer o comportamento do sinal de entrada, o erro quadrático médio dos algoritmos de treinamento e suas taxas de convergência.

Palavras chaves: Sistema Autoregressivo (AR), mapa logístico, função de base radial, *least squares, improved least squares*, sistema caótico.

ABSTRACT

In this work we show the operation of blind equalization using the artificial neural network, radial base function (RBF). By exploring the characteristics of the input signal, the neural network is used to predict the expected output of the reverse filter (equalization filter). Equalizer coefficients are obtained by using least squares (LS) and improved least squares (ILS) algorithms, the latter being superior to the first one in white noise environments. The simulation results shown include the ability of RBF to predict the input signal behavior, the mean squared error of the training algorithms and their convergence rates.

Key words: Autoregressive System (AR), logistic map, radial base function, least squares, improved least squares, chaotic system.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

PMC: *Perceptron* de multicamadas.
RBF: função de base radial.
LS: *least square*.
ILS: *improved least square.*
MNPE: *minimum non linear preditor error*.
ISI: interferências entre símbolo.
LMS: *least minimum square*.
RLS: *recursive least square.*
MSE: *minimum squares error*
η: taxa de aprendizagem.
$\boldsymbol{\varepsilon}$: erro.
$\boldsymbol{\varphi}(.)$: função de ativação.
δ: gradiente local.
$\boldsymbol{\varrho}$: erro médio.
$\boldsymbol{\partial}$: diferencial.
Θ: *bias.*
$\boldsymbol{\zeta}$: erro de aproximação.
k: centros (k = *means*).
$g(.)$: função de base radial.
h^T: coeficientes do sistema.
$\hat{h}^T$: Coeficientes do filtro inverso.
$\boldsymbol{\lambda}$: parâmetro de controle.
$\sum$: somatório.
σ^2: desvio padrão.

CAPÍTULO 1: INTRODUÇÃO

1.1. REDES NEURAIS

A ideia de se construir uma máquina ou mecanismo autônomo, que seja dotado de inteligência, se constitui um sonho antigo dos pesquisadores das áreas de ciências e engenharias. Embora os primeiros trabalhos em redes neurais artificiais (RNA) tenham sido publicados há mais de 50 anos, tal tema começou a ser fortemente pesquisado a partir do início dos anos 1990, tendo ainda um potencial de pesquisa imenso [1].

A era moderna das redes neurais começou com o trabalho pioneiro de McCulloch e Pitts em 1943. McCulloch foi um psiquiatra e neuroanatomista por treinamento; passou cerca de 20 anos refletindo sobre a representação de um evento no sistema nervoso, Pitts foi prodígio matemático que associou a McCulloch em 1942. De acordo com Rall em 1990, o artigo de 1943 de McCulloch e Pitts surgiu dentro de uma comunidade de modelagem neural que tinha estado em atividade na *University of Chicago* por pelo menos cinco anos antes de 1943, sob a liderança de Rashevsky.

/

No seu clássico artigo, McCulloch e Pitts descrevem um cálculo lógico das redes neurais que unificava os estudos de neurofisiologia e da lógica matemática. Eles assumiam que o seu modelo formal de um neurônio seguia uma lei "tudo ou nada". Com um número suficiente dessas unidades simples e com conexões sinápticas ajustadas apropriadamente e operando de forma síncrona, McCulloch e Pitts mostraram que uma rede assim constituída realizaria, a princípio, aproximação de qualquer função computável. Este era um resultado muito significativo e som ele é geralmente aceito o nascimento das disciplinas de redes neurais e inteligência artificial.

O artigo de 1943 de McCulloch e Pitts foi amplamente lido naquele tempo e ainda é. Ele influenciou von Neumann a usar chaves de atraso idealizadas, derivadas do neurônio de McCulloch e Pitts na construção do EDVAC (*Electronic Discrete Variable Automatic Computer*) que foi desenvolvido a partir do ENIAC (*Electronic Numerical Integrator and Computer*) [2]. O ENIAC foi o primeiro computador eletrônico de propósito geral, que foi construído na Escola de Engenharia Elétrica *Moore* da *University of Pennsylvania* de 1943 a 1946. A teoria de McCulloch-Pitts sobre redes neurais formais se destacou de forma proeminente na segunda das quatro palestras proferidas por von Neumann na University of Illinois em 1949 .

Em 1948, foi publicado o famoso livro *Cybernetics* de Wiener, descrevendo alguns conceitos importantes sobre controle, comunicação e processamento estatístico de sinais. A segunda edição do livro foi publicada em 1961, adicionando material novo sobre aprendizagem e auto-organização. No capítulo 2 de ambas edições desse livro, Wienner parece compreender o significado físico da mecânica estatística no contexto desse assunto, mas foi com Hopfield (mais de 30 anos depois) que se conseguiu consumar a ligação entre a mecânica estatística e os sistemas de aprendizagem .

O próximo desenvolvimento significativo das redes neurais veio em 1949, com a publicação do livro de Hebb *the Organization of Behavior*, no qual foi apresentada pela primeira vez uma formulação explícita de uma regra de aprendizagem fisiológica para modificação sináptica. Especificamente, Hebb propôs que a conectividade do cérebro é continuamente modificada conforme um organismo vai aprendendo tarefas funcionais diferentes e que agrupamentos neurais são criados por tais modificações. Hebb deu seguimento a uma sugestão anterior de Ramón y Cajál e apresentou o seu agora famoso *postulado de aprendizagem*, que afirma que a eficiência de uma sinapse variável entre dois neurônios é aumentada pela ativação repetida de um neurônio causada pelo outro neurônio, através daquela sinapse. O livro de Hebb foi imensamente influente entre os psicólogos, mas lamentavelmente ele teve pouco ou nenhum impacto sobre a comunidade de engenharia .

O livro de Hebb tem sido uma fonte de inspiração para o desenvolvimento de modelos computacionais de *sistemas de adaptativos e de aprendizagem*. O artigo [3] talvez seja a primeira tentativa de usar

simulação computacional para testar uma teoria neural bem-formulada com base no postulado de aprendizagem de Hebb; nos resultados de simulação relatados naquele artigo mostram claramente que se deve adicionar inibição para que teoria realmente funcione. Naquele mesmo ano, [4] demonstrou que uma rede neural com sinapses modificáveis pode aprender a classificar conjuntos simples de padrões binários em classes correspondentes. Uttley introduziu o assim chamado neurônio integra e dispara com fuga, o qual foi mais tarde analisado formalmente por Caianiello [5]. Em um trabalho posterior, Uttley [6] formulou a hipótese de que a eficiência de uma sinapse variável do sistema nervoso depende da relação estatística entre os estados flutuantes em ambos os lados daquela sinapse, fazendo assim uma associação com a teoria da informação de Shannon.

Em 1952, foi publicado o livro de Ashby, *Design for a Brain: The Origin of Adaptive Behavior*, que é tão fascinante de ser lido hoje em dia como deve tê-lo sido naquela época. O livro trata da noção básica de que o comportamento adaptativo não é inato, mas sim é aprendido, e que através da aprendizagem o comportamento de um animal (sistema) normalmente muda para melhor. O livro enfatiza os aspectos dinâmicos do organismo vivo como uma máquina e o conceito correlacionado de estabilidade.

Em 1954, Minsky escreveu uma tese de doutoramento em "redes neurais" na University of Princeton, intitulada "*Theory of Neural-Analog Reinforcement Systems and Its Aplication to the Brain-Model Problem*". Em 1961, foi publicado um artigo excelente de Minsky sobre Inteligência Artificial (IA) intitulado "*Steps Toward Artificial Intelligence*"; este artigo contém uma grande seção sobre o que agora é denominado redes neurais. Em 1967, foi publicado o livro de Minsky, *Computation: Finite and Infinite Machines*. Este livro, escrito de forma clara, estendeu os resultados de 1943 de McCulloch e Pitts e os colocou no contexto da teoria dos autômatos e da teoria da computação [1].

Também em 1954, a ideia de um filtro adaptativo não-linear foi proposta por Gabor, um dos pioneiros da teoria da comunicação e o inventor da holografia. A aprendizagem era realizada alimentando-se a máquina com amostras de um processo estocástico, juntamente com a função-alvo que a máquina deveria produzir .

Nos anos 50, iniciou-se o trabalho sobre a *memória associativa* por Taylor em 1956. Ele foi seguido por Steinbruch [7] que introduziu a matriz de aprendizagem; esta matriz consiste de uma rede planar de chaves interpostas entre arranjos de receptores "sensoriais" e atualizadores "motores". Em 1969, foi publicado por Willshaw, Buneman e longguet-Higgins um elegante artigo sobrea memória associativa não-holográfica. Este artigo apresenta dois modelos engenhosos de rede: um sistema ótico simples realizando uma memária de correlação e uma rede neural intimamente relacionada com ele, inspirada na memória óptica. Outras contribuições significativas ao desenvolvimento inicial da memória associativa incluem os artigos de [8], [9] e [10], que de maneira independente e no mesmo ano introduziram a ideia de uma *memória por matriz de correlação*, baseada na regra de aprendizagem do produto externo.

Cerca de 15 anos após a publicação do clássico artigo de McCulloch e Pitts, uma nova abordagem para o problema de reconhecimento de padrões foi introduzida por Rosenblatt [11] em seu trabalho sobre o *perceptron*, um método inovador de aprendizagem supervisionada. O coroamento do trabalho de Rosenblat foi o chamado *teorema da convergência do perceptron*, cuja primeira demonstração foi delineada por Rosenblatt [12]; outras provas do teorema também apareceram em Novikoff [13] e outros. Em 1960, Widrow e Hoff introduziram o algoritmo do mínimo quadrático médio (*LMS, Least Mean-Square*) e usaram para formular o Adaline está no procedimento de aprendizagem. Uma das primeiras redes neurais em camadas treináveis com múltiplos elementos adaptativos foi a estrutura Madaline (*muti-adaline*) proposta por Widrow e seus estudantes [14]. Em 1967, Amari utilizou o método do gradiente estocástico para classificação adaptativa de padrões. Em 1965, foi publicado o livro de Nilson, *Learning Machines* [15] que ainda é a exposição mais bem escrita sobre padrões linearmente separáveis por hipersuperfícies. Durante o período clássico do perceptron nos anos 1969, parecia que as redes neurais poderiam realizar qualquer coisa. Mas então veio o livro de Minsky e Papert [16], que utilizaram a matemática para demostrar que existem limites fundamentais para aquilo que os perceptrons de camada única podem calcular. Em uma breve seção sobre perceptrons de multicamadas, eles afirmam que não havia razão para supor que qualquer uma das limitações do perceptron de camada única poderia ser superada na versão multicamadas.

Um problema importante encontrado no projeto de um perceptron de multicamadas é o problema de atribuição de crédito, quer dizer, o problema de atribuir crédito a neurônios ocultos da rede. A terminologia "atribuição de credito" foi utilizada primeiro por Minsky [17], sobre o título de "O problema de atribuição de credito para sistemas de Aprendizagem por Reforço ". No final dos anos 1960, já havia sido formulado a maioria das ideias e conceitos necessários para resolver o problema de atribuição de crédito do perceptron, bem como muitas das ideias que fundamentam as redes neurais de atratores recorrentes que são agora denominadas redes de Hopfield. Entretanto, tivemos que esperar até os anos 80 para que emergissem as soluções para esses problemas clássicos. De acordo com Cowan [18] houve três razões para este atraso de mais de 10 anos: uma razão foi tecnológica, outra razão foi em parte psicológica e em parte financeira e em fim a analogia entre redes neurais e *spins* de grade foi prematura.

Uma atividade importante que emergiu nos anos 70 foram os mapas- autos-organizáveis utilizando aprendizagem competitiva. O trabalho em simulação computacional feito por Von der Malsburg [19] talvez tenha sido primeiro a demonstrar a auto-organização. Em 1976, Willshaw e Von der Malsburg publicaram o primeiro artigo sobre a formação de mapas-auto-organizáveis, motivados pelos mapas ordenados de forma topológica de cérebro.

Nos anos 80, foram feitas importantes contribuições em várias frente à teoria e ao projeto de redes neurais, e com isso houve um ressurgimento do interesse pelas redes neurais .

1.1.1. Características principais.

As características mais relevantes envolvidas com aplicações de redes neurais artificiais são:

- Os parâmetros internos da rede, chamados de pesos sinápticos, são ajustados a partir de um treinamento sucessivo de padrões, amostras ou medidas relacionadas ao comportamento do processo.
- Capacidade de aprendizado: após um método de treinamento, a rede tem capacidade de fazer o relacionamento que existe entre a entrada e a saída.
- Organização de dados: baseado em característica similar de um conjunto de informações a respeito de um processo, a rede interna consegue agrupar os padrões que apresentam elemento em comum
- Facilidade de implementar: após o processamento de treinamento, os seus resultados são normalmente obtidos por algumas operações matemáticas elementares.

1.1.2. Neurônios biológicos

O cérebro humano aqui é visto como uma rede neural que tem capacidade de receber informações exteriores, de processa-las e de dar resposta. Algo que fazemos na dia a dia. Por exemplo, pelos olhos, conseguimos distinguir as cores das coisas. O esforço para entender o cérebro se tornou mais fácil pelo trabalho pioneiro de Ramón y Cajál [20], que introduziu a ideia dos neurônios como constituintes estruturais do cérebro humano. O cérebro compensa a taxa de operação relativamente lenta de um neurônio pelo número realmente espantoso de células nervosas, com conexões maciças entre-se. Estima-se que haja aproximadamente 10 bilhões de neurônios no córtex humano e 60 trilhões de sinapses ou conexões [21]. O cérebro é uma estrutura extremamente eficiente, a eficiência energética do cérebro é de aproximadamente 10^{-16} *joules* por operação por segundo [22].

As sinapses são organizadas e funcionais para interligar os neurônios entre eles. Há vários tipos de sinapses, porém a mais comum é a sinapse química. Nosso objetivo não é falar profundamente sobre sinapse química, no entanto é mostrar para os leitores de onde veio a ideia dos neurônios artificias. Para quem quiser entender melhor, sugerimos o livro do *Shepherd e Kosh* [21] para vocês. A seguir, há a Figura *1.1* que mostra a estrutura de um neurônio :

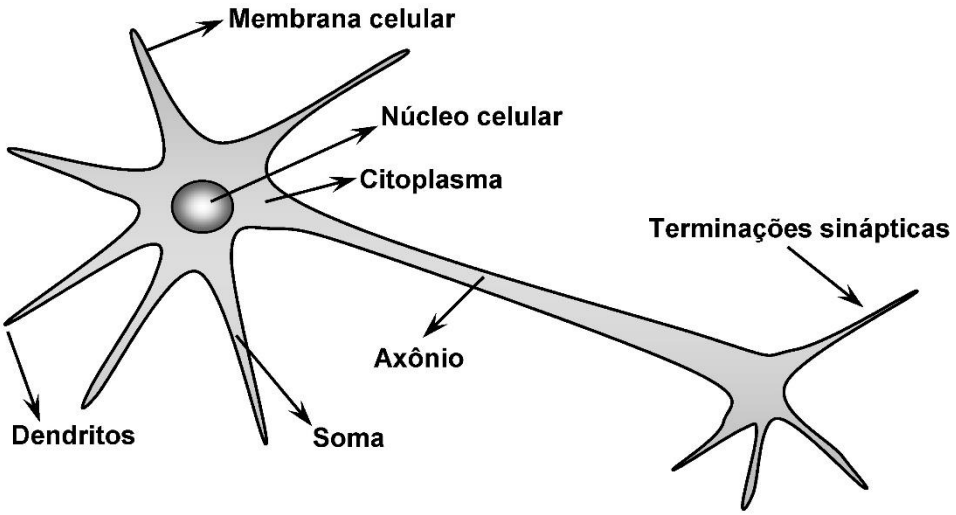

Figura 1.1. Neurônios biológicos [26]

1.1.3. Neurônios artificiais.

Como foi dito no início, não podemos estudar a rede neural artificial sem falar dos neurônios biológicos, pois a estrutura das redes neurais artificiais foi desenvolvida a partir dos modelos conhecidos de sistemas nervosos biológicos e do próprio cérebro humano. Os neurônios artificias são modelados de acordo com os neurônios biológicos.

Os neurônios artificiais utilizados nos modelos de redes artificiais são não lineares, fornecem saídas tipicamente continuas, recebem uma entrada, fazem cálculo em relação a sua função de ativação apresentam resultados. O modelo de neurônio mais simples e que engloba as principais características de uma rede neural biológico, isto é, paralelismo e alta conectividade, foi proposto por McCulloch & Pitts [23], sendo ainda o modelo mais utilizado nas diferentes arquiteturas de redes neurais artificiais.

Nessa seção, mostraremos como funciona cada neurônio da rede. Ele recebe informação vindo do exterior, que representaremos pelo conjunto $\{x_1, x_2, x_3 \ldots x_p\}$ são análogas aos impulsos elétricos captados pelos dendritos no neurônio biológico.

A ponderações exercidas pelas junções sinápticas do modelo biológico são representadas no neurônio artificial pelo conjunto de pesos sinápticos $\{w_1, w_2, w_3 \ldots w_p\}$. Cada uma das entradas $\{x_i\}$ do neurônio é então executada por meio de suas multiplicações pelos respectivos pesos sinápticos $\{w_i\}$ ponderando-se, isso será feita para todos os sinais externos que chegam ao neurônio. Assim, a saída do neurônio será a soma ponderada de seus sinais de entradas, denotado por U.

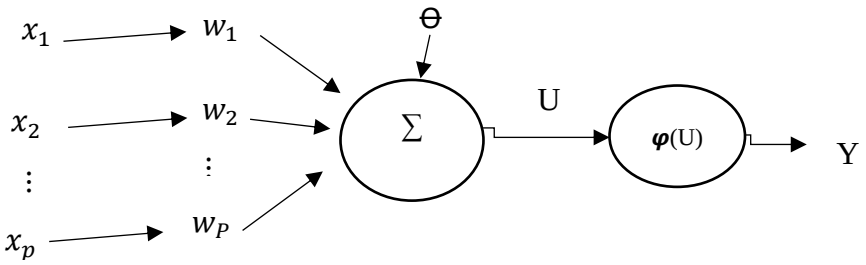

Figura 1.2. Modelo de um Neurônio artificial [22].

$$U(t) = \sum_{i=1}^{n} x_i w_i(t) + \Theta \quad (1.1)$$

$$Y(t) = \varphi(U(t)) \quad (1.2)$$

Como pode ser visto pela Fig. (1.2), temos sete (7) elementos principais, que nós vamos detalhar agora.

a) Sinais de entradas: $\{x_1, x_2, x_3 \ldots x_p\}$, são variáveis que receberão os valores de aplicação especifica e vêm do sistema exterior. Eles variam de um sistema para outros.
b) Pesos: $\{w_1, w_2, w_3 \ldots w_p\}$, são valores que serão ponderados com os sinais de entradas.
c) Símbolo sigma $\{\sum\}$: somatório ponderados.
d) *Bias* $\{\Theta\}$: ajuda orientar o valor do item *c* na direção da saída do neurônio. Pode assumir um valor negativo ou positivo.
e) Valor de ativação {U}: é a função de ativação.
f) Função de ativação $\{\boldsymbol{\varphi}\}$: seu objetivo é limitar a saída do neurônio dentro de um intervalo de valores razoáveis a serem assumidos pela sua própria imagem funcional
g) Sinal de saída {Y}: é o resultado final de um neurônio, que pode ser utilizado para outro neurônio como sinal de entrada quando estão sequencialmente interligados.
   As duas expressões seguintes sintetizam o resultado produzido pelo neurônio artificial por McCulloch e Pitts, ou seja:

Com todo esse detalhe podemos resumir assim o funcionamento de um neurônio artificial por seguintes passos [22]:

- Precisa ter um conjunto de valores que representam as variáveis de entrada do neurônio;
- Multiplicação de cada entrada pelo seu respectivo peso sináptico;
- Somar todos valores do item anterior para obter um potencial de ativação;
- Escolher uma função de ativação apropriada com objetivo de limitar a saída do neurônio
- Por fim, calcular a saída do neurônio, aplicando o potencial de ativação dentro da função de aplicação escolhida.

Os diferentes tipos de neurônios dependem da sua função de ativação. As principais funções são: função degrau, função bipolar, função rampa simétrica, função logística, função tangente hiperbólica, gaussiana e a função linear.

Vamos enfatizar mais sobre a função degrau e a função gaussiana. Uma é parcialmente diferenciável e outro totalmente diferenciável.

a) Função degrau

O resultado dessa função assumirá o valor unitário. 1 ou 0.

$$\boldsymbol{\varphi}(.) = \begin{cases} 1, se\ U \geq 0 \\ 0, se\ U < 0 \end{cases} \quad (1.3)$$

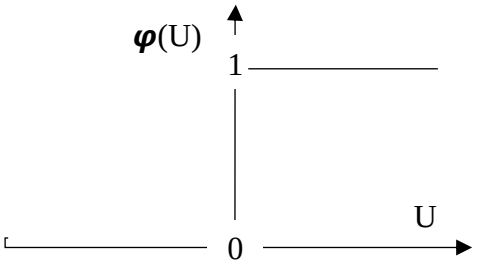

Figura 1.3: Gráfico da função grau

b) Função Gaussiana:

Em relação à utililização da função de ativação gaussiana, a saída do neurônio produzirá resultados iguais para aqueles valores de potencial de ativação **{*U*}** que estejam posicionados a uma mesma distância de seu centro(média), sendo que a curva é simétrica em relação a este. A função gaussiana é dada por:

$$\varphi\ (\mathrm{U}) = e^{-\frac{(u-c)^2}{2\sigma^2}} \tag{1.4}$$

Em que *c* é um parâmetro que define o centro da função gaussiana e σ denota o desvio padrão associado a mesma, isto é, o quão dispersada está a curva em relação ao seu centro. A representação gráfica desta função é ilustrada na Fig.(1.4).

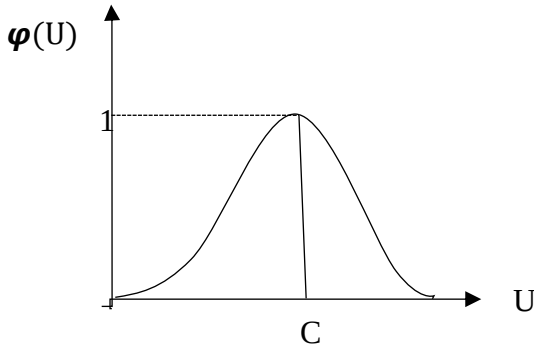

Figura 1.4. Função de ativação Gausiana

1.2. PROBLEMA DE EQUALIZAÇÃO CEGA USANDO REDES NEURAIS

Nos últimos anos, a arte de se usar redes neurais artificiais para comunicações sem fio ganhou muito impulso. Os equalizadores lineares geralmente usam filtros lineares com estrutura transversal ou de rede com algoritmo de adaptação como por exemplo, o mínimo quadrado recursivo (RLS), mínimo quadrático médio (LMS), RLS rápido, RLS de raiz quadrada, RLS de gradiente, etc. No entanto, equalizadores lineares não funcionam nem com canais com nulos espectrais [4].

Redes neurais artificias são capazes de formar limites de decisão arbitrariamente não lineares para assumir tarefas complexas de classificação [4]. Esse trabalho mostra o funcionamento da função de base radial, uma das técnicas de redes neurais artificiais para a modelagem do fenômeno não linear de equalização de canal de comunicação.

A equalização refere-se a qualquer processamento de sinal. Técnica utilizada no receptor para combater a interferência Inter símbolo (ISI). Em canais dispersivos, as técnicas padrão de equalização começam pela modelagem de um canal de comunicação como um filtro adaptativo com uma função de transferência. O equalizador, que faz parte do receptor, então estima os parâmetros do filtro inverso.

Para desfazer os efeitos desta distorção de canal variável no tempo [4], o equalizador extrai o sinal desejado aplicando algoritmo adaptativo usando redes neurais artificiais, que minimiza o erro entre a saída do equalizador (filtro inverso) e o sinal de teste atrasado.

1.3. POTENCIAS ÁREAS DE APLICAÇÕES.

As aplicações que envolvem a utilização de sistemas considerados inteligentes são as mais variadas possíveis, por exemplo [22]:

- Avaliação de imagem captadas por satélite;
- Classificação de padrões de escrita e de fala;
- Reconhecimento de faces em visão computacional;
- Previsão de ações no mercado financeiro;
- Identificação de anomalias em imagens médicas;
- Identificação automática de perfis de crédito para clientes de instituições financeiras;
- Equalização cega.

No artigo [24], mostra que na área de química, as redes neurais artificiais são utilizadas para obtenção novos compostos poliméricos.

A área de biologia usa as redes neurais artificiais com objetivo de se identificar espécies de morcegos a partir de seus sinais de eco localização (biosonar) emitidos durante os voos [25].

O ramo da ecologia se beneficia da capacidade das redes neurais artificiais em extrair informações, sendo possível realizar análises entre a influência do clima atual frente à taxa de crescimento de árvores

A habilidade das redes neurais artificiais em classificar padrões pode ser observada até mesmo no ramo da etologia, no qual se busca a diferenciação entre as diversas expressões faciais que caracterizam os sentimentos humanos. Outra aplicação relacionada à classificação de padrões é dissertada em que redes neurais artificiais são usadas para classificar fontes de correntes harmônicas em sistemas de distribuição de energia elétrica.

A indústria de alimentos também tem sido beneficiada com a utilização de redes neurais artificiais, tal como aquela usada na classificação de diversas variedades existentes de chá Outra abordagem em rede neural foi abordada para implementar uma ressonância magnética nuclear visando à classificação de sexo e raça dos animais [22].

No setor automotivo e aeroespacial encontram-se aplicações de redes neurais artificiais para auxiliar no mapeamento de processos que envolvem estimativas de variáveis de controle e parâmetros de projeto. Como alguns destes exemplos, foram propostos esquemas de modelagem e estratégias de controle para veículos aéreos não-tripulados .

1.4. OBJETIVO DO TRABALHO E ORGANIZAÇÃO

Aqui nesse trabalho tratamos de uma nova abordagem de um equalizador cego baseado sobre a rede neural artificial de base radial. A RBF (Função de base radial) é usada como preditor para achar os coeficientes da saída do filtro inverso. Temos dois casos, o primeiro caso é um meio sem ruído e o segundo um meio com ruído. Cada caso tem uma abordagem diferente em relação as técnicas de recuperação do sinal. Para um meio sem ruído usamos o LS (*least square*) e para o meio com ruído usamos o ILS (*improved least square*). No capítulo 4 explicaremos com detalhe cada um desses casos.

A motivação desse estudo é devido ao seu uso em várias áreas, por exemplo, na área militar, é usado na intercepção de um sinal incógnito. Isso ajudará para recuperar a mensagem dos inimigos e prevenir os ataques imprevisíveis. Há também na área de comunicação, como, controle e processamento de sinais.

Neste trabalho falaremos da importância de usar uma rede neural artificial de base radial para resolver o problema de equalização cega. A RBF é usada como preditor para achar os coeficientes do filtro inverso no receptor por meio da deconvolução.

Este trabalho é dividido como seguinte: capítulo 2, mostra como é feito uma rede neural artificial, a forma e seus diversos arranjos. Fala do funcionamento do *Perceptron* de única e multicamada e suas estruturas e diferencias. Também inicia a implementação do algoritmo de *backpropagation*. No capítulo 3 é implementado a função de base radial, que é a técnica usada neste trabalho para recuperar o sinal de entrada. No capítulo 4 fala-se da equalização cega usando RBF como preditor. E no último capítulo é a conclusão do trabalho e proposta para o próximo trabalho e pesquisa.

1.5. RESUMO

Vimos o histórico das redes neurais artificiais e suas respectivas áreas de aplicação. No próximo capitulo vamos aprender as arquiteturas e os algoritmos de aprendizagem das redes neurais artificiais.

CAPÍTULO 2: ARQUITETURAS DE REDES NEURAIS ARTIFICIAIS E PROCESSOS DE TREINAMENTO

2.1. INTRODUÇÃO

Esse capítulo mostra como é feito uma rede neural artificial, a forma e seus diversos arranjos ou disposição, uns aos outros. Esses arranjos são essencialmente estruturados através do direcionamento das conexões sinápticas dos neurônios.

Já a topologia de uma rede neural, considerando determinada arquitetura, pode ser definida como sendo as diferentes formas de composições estruturais que esta poderá assumir. Por exemplo, uma rede neural pode ter 10 neurônios e outra 20 neurônios, e dentro uma rede neural artificial pode haver diferentes funções de ativações [22].

Por outro lado, o treinamento de uma arquitetura específica consiste na aplicação de um conjunto de passos ordenados com o intuito de ajustar os passos e os limiares (*bias*) de seus neurônios. Assim, o tal processo de ajuste, também conhecido como algoritmo de aprendizagem, tem como objetivo sintonizar a rede para que as suas respostas estejam próximas dos valores desejados.

2.2. PRINCIPAIS ARQUITETURAS DE REDES NEURAIS ARTIFICIAIS

Basicamente uma rede neural tem três diferentes partes, que são nomeadas de:

- **Camada de entrada**: é uma parte da rede que recebe os sinais (amostras) a partir de um meio externo.
- **Camada escondida**: ou camadas intermediárias, ocultas ou invisíveis. São aquelas compostas de neurônios que possuem a responsabilidade de extrair as características associadas ao processo ou sistema a ser inferido. Quase todo o processamento interno da rede é realizado nessas camadas.
- **Camada de saída**: é também constituída de neurônios, sendo responsável pela produção e apresentação dos resultados finais da rede, os quais são advindos dos processamentos efetuados pelos neurônios das camadas anteriores.

As principais arquiteturas de redes neurais artificiais, considerando a disposição de seus neurônios, assim como suas formas de interligação entre eles e a constituição de suas camadas, podem ser divididas em: redes *feedforward* (alimentação à frente) de camada simples, redes *feedforward* de camadas múltiplas, redes recorrentes e redes reticuladas .

2.2.1. Arquitetura de camada simples:

Só tem neurônios na camada intermediaria. A camada de entrada é diretamente ligada aos neurônios da camada intermediária, que, por sua vez, está conectada diretamente com a saída. Seu fluxo de informação é unidirecional, quer dizer, da camada de entrada em direção à camada de saída. Essas redes são tipicamente empregadas em problemas envolvendo classificação de padrões e filtragem linear. A partir ainda da análise da Fig. (2.1), observa-se então que a quantidade de saídas nas redes pertencentes à arquitetura sempre coincidirá com o número de neurônios.

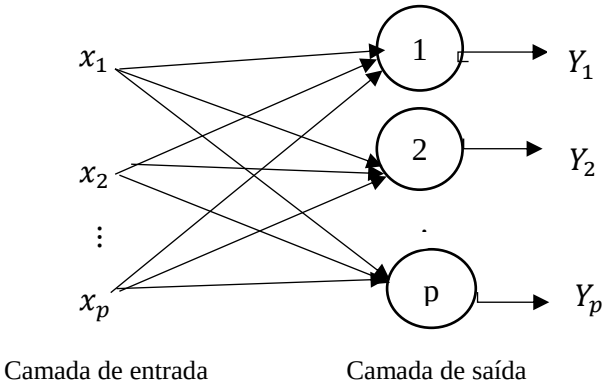

Figura 2.1. Arquiteta de única camada

2.2.2. Arquiteturas de multicamadas

Essa arquitetura de redes neurais possui mais de uma camada de neurônios. há camada de entrada, camada intermediária há uma ou mais camadas, e há camada de saída. As saídas de certos neurônios constituem entradas de outros neurônios. A Fig. 2.2 mostra essa arquitetura. Entre os principais tipos de redes com arquitetura de camadas múltiplas se encontram o *Perceptron* multicamadas e as redes de base radial, cujos algoritmos de aprendizado utilizados em seus processos de treinamento são respectivamente baseados na regra generalizada e na regra delta, conforme os capítulos posteriores [22].

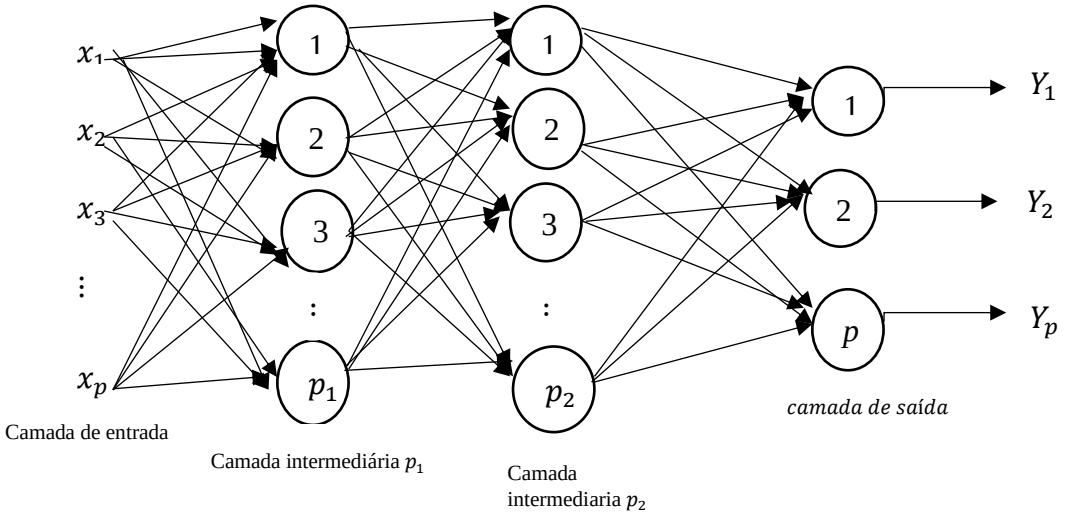

Figura 2.2. Arquiteturas de camada múltiplas

Como mencionado acima, há várias arquiteturas, mas explicamos somente as duas que usaremos mais para frente.

2.3. PROCESSOS DE TREINAMENTO

Um aspecto importante das redes neurais artificiais está na sua capacidade de aprender a partir de amostras apresentadas na entrada que exprimem o comportamento do sistema, em seguida, após a rede ter aprendido o relacionamento entre as entradas e saídas, esta é capaz de generalizar. A rede será então capaz de produzir uma saída próxima daquela desejada a partir de quaisquer sinais inseridos em suas entradas.

Portanto, processo de treinamento de uma rede neural consiste na aplicação de passo ordenados que sejam necessários para sintonização dos pesos sinápticos e limiares de seus neurônios, tendo-se como objetivo final a generalização de soluções a serem produzidas pelas suas saídas, cujas respostas são representativas do sistema físico em que estas estão mapeando.

O conjunto desses passos ordenados visando o treinamento da rede é denominado algoritmo de aprendizagem. Ao longo de sua aplicação, a rede será capaz de extrair características discriminantes do sistema a ser mapeado por intermédio de amostras que foram retiradas de seu contexto. O conjunto total das amostras disponíveis sobre o comportamento do sistema é dividido em dois subconjuntos, os quais são denominados de subconjunto de treinamento e subconjunto de teste. O subconjunto composto totalmente de 60 a 90 % das amostras do conjunto total, será usado para aprendizado da rede. Já o subconjunto de teste, cuja composição está entre 40 a 10% do conjunto total de amostras serão utilizados para verificar o funcionamento do sistema e sua validação. [22]

Durante o processo de treinamento de redes neurais artificiais, cada apresentação completa das amostras pertencentes ao subconjunto de treinamento visando, sobretudo, o ajuste dos pesos sinápticos e limiares de seus neurônios, será denominada de época de treinamento. No próximo capítulo quando falarmos de algoritmo de aprendizagem de cada modelo, explicaremos melhor sobre subconjunto de treinamento, ajusto de pesos e época de treinamento.

2.3.1. Treinamento supervisionado.

Treinamento supervisionado consiste em se ter disponível uma entrada e uma saída correspondente. Geralmente costumamos chamar de amostras, ou padrão que é uma entrada e uma saída. Desta forma, há então a necessidade de se disponibilizar uma tabela de dados (entradas, saídas) representativa do processo, também conhecida por tabela atributos/valores, e que contemple inclusive o seu comportamento, pois é a partir de tais informações que as estruturas neurais formularão as " hipóteses" sobre aquilo a ser aprendido .

Neste caso, aplicação do treinamento supervisionado depende apenas da disponibilidade dessa tabela, que tem o valor da entrada e da saída já conhecido a priori. É um treinamento que se comporta como se houvesse um professor durante seu ajuste de peso, ele terminará somente quando a saída será igual ou próxima a sua saída dada no começo do treinamento.

Os pesos sinápticos e limiares são continuamente ajustados mediante a aplicação de ações comparativas, executadas pelo próprio algoritmo de aprendizagem, que supervisionam a defasagem entre as respostas produzidas pela rede em relação àquelas desejadas, sendo esta diferença usada no procedimento de ajuste. A rede será considerada treinada quando a tal defasagem estiver dentro de valores aceitáveis, levando-se em consideração os propósitos de generalização de soluções .

2.3.2. Treinamento não-supervisionado.

Diferentemente do treinamento supervisionado, durante a aplicação de um algoritmo de aprendizado baseado em treinamento não-supervisionado não existe as respectivas saídas desejadas. Lembrando que, no algoritmo supervisionado, há uma entrada e saída correspondente, ou seja, o problema a ser resolvido já te oferece as saídas, mas na aprendizagem não supervisionada, não tem saídas, há somente entradas.

Consequentemente, a própria rede deve se auto organizar em relação às particularidades existentes entre os elementos componentes do conjunto total de amostras, identificando subconjuntos que contenham similaridades. Os pesos sinápticos e limiares dos neurônios da rede são então ajustados pelo algoritmo de aprendizado de forma a refletir esta representação internamente dentro da própria rede.

Todavia, este projeto abordará somente a aprendizagem supervisionada por conta da aplicação explorada.

2.4. FUNCIONAMENTO DO *PERCEPTRON*

Como pode ser visto na Fig. (2.1), a arquitetura de *Perceptron* de única camada é bastante simples. As entradas apresentadas no sistema as informações vindo do exterior, cada entrada será

associado a um peso et depois somado a um *bias* para gerar o que chamamos de sinal de ativação. Depois, aplicamos esse sinal na função que geralmente é a função degrau que explicamos acima. Nesse trabalho só usaremos a função de grau. Em relação a saída produzida, 1 ou 0 será classificado em uma das regiões.

Em termos matemáticos, o processamento interno realizado pelo *Perceptron* pode ser descrito pelas seguintes expressões:

$$U(t) = \sum_{i=1}^{n} x_i . w_i(t) - \theta \tag{2.1}$$

$$Y(t) = \varphi(U(t)) \tag{2.2}$$

O Perceptron de única camada serve para separar padrões em duas classes ou grupos diferentes. Mas primeiramente esses padrões devem ser linearmente separáveis.

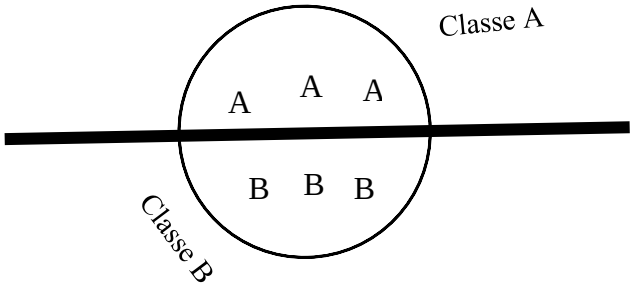

Figura. 2.3. Linha de fronteira de um classificador de padrões de duas classes.

Olhando a Fig. 2.3. Vemos uma linha que separa duas classes A e B. Nosso objetivo é de achar ou descrever a equação matemática dessa linha ou reta.

2.5. PROCESSO DE TREINAMENTO DO *PERCEPTRON* DE ÚNICA CAMADA

Antes de falar sobre o treinamento do Perceptron de única camada, vamos ilustrar o funcionamento de um pequeno exemplo primeiro. Suponha que temos um sistema com somente duas entradas. Ele funcionará como um classificador, quando a função de ativação for maior ou igual a zero, a saída irá por um lado, se for menor que zero, a saída irá por outro lado.

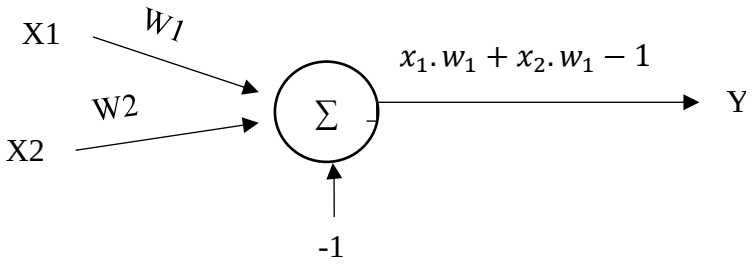

Figura 2.4. Modelos do Perceptron com duas entradas [22].

Suponhamos que as variáveis de entrada do Perceptron se originem de duas classes linearmente separáveis. Seja T_1 o subconjunto de vetores de treinamento, $x_1(1), x_1(2)$, ... que pertence à classe A e seja T_2 o subconjunto de vetores de treinamento, $x_2(1), x_2(2), \ldots$ que pertence à classe B. A união de classe T_1 *e* T_2 é o conjunto de treinamento completo T. Dados os conjuntos de vetores T_1 *e* T_2 para treinar o classificador, o processo de treinamento envolve o ajuste do vetor de peso $\boldsymbol{w}$ de tal forma que

as duas classes A e B sejam linearmente separáveis. Isto é, existe um vetor de peso $\boldsymbol{w}$ para o qual podemos afirmar:

$$\begin{aligned}&(w_1, w_{2,\dots} w_p)x > 0 \text{ Para todo vetor de entrada } \mathbf{x} \text{ pertencente à classe A}\\&(w_1, w_{2,\dots} w_p)x \leq 0 \text{ Para todo vetor de entrada } \mathbf{x} \text{ pertencente à classe B}\end{aligned} \tag{2.3}$$

O problema de treinamento de para o Perceptron elementar é, então, encontrar um vetor de peso $\boldsymbol{w}$ tal que as duas desigualdades da Eq. (2.3) sejam satisfeitas. O algoritmo de aprendizagem pode ser descrito como segue:

1. Identificar as entradas e suas respectivas saídas. Se os pesos levam diretamente à saída correspondentes, então não precisa atualizar o peso de acordo com a regra:

$$\boldsymbol{w(t+1) = w(t)} \tag{2.4}$$

2. Caso os pesos não levem para as saídas correspondentes, então os pesos serão atualizados de acordo com a regra delta:

$$\boldsymbol{w(t+1) = w(t) \pm \eta \varepsilon(t) x} \tag{2.5}$$

Em que:

η: Representa o passo de aprendizagem, geralmente varia entre 0 e 1;
ε: erro, que é a diferença entre a saída dada e saída calculada;
x: valor de entra.

O algoritmo de aprendizagem ficará dentro de um controle até que a condição seja satisfeita, que é aproximação da saída calculada com a saída dada. Com este procedimento, constatamos que se o produto interno $(w_1, w_{2,\dots} w_p)x$ na iteração passada tiver errado, ou seja, não leva à saída dada, então $(w_1, w_{2,\dots} w_p)x$ na iteração seguinte terá uma saída correta, dada. Em outras palavras, cada padrão é apresentado repetidamente ao Perceptron até que aquele padrão seja classificado corretamente. Independentemente do valor atribuído a $\boldsymbol{w}$ como inicial, a convergência do Perceptron está assegurada.

Vamos resumir a convergência do *perceptron*:

1. Inicializamos os pesos como um valor arbitrário; $w_{(t)}$= 0
2. Associa o vector de entrada *x(t)* com sua saída respectiva dada *d(t)*
3. Calcule a resposta do Perceptron, *Y(U)*
4. Atualize os pesos do Perceptron conforme a Eq. (2.5)
5. Incremente o passo de tempo t até satisfazer a condição de parar.

2.6. PROCESSO DE TREINAMENTO DO *PERCEPTRON* MULTICAMADAS

Contrariamente ao Perceptron de única camada, o Perceptron multicamadas é composto de uma camada de entrada, uma ou mais camadas intermediárias e uma camada de saída. O sinal de entrada se propaga para frente através da rede, camada por camada. Estas redes neurais são normalmente chamadas *de Perceptron de múltiplas camadas* (PMC, *multilayer perceptron*), as quais representam uma generalização do perceptron de única camada mencionado na seção anterior.

Os perceptrons de multicamadas têm sido aplicados com sucesso para resolver diversos problemas difíceis através do seu treinamento de forma supervisionada, com um algoritmo muito popular, conhecido como algoritmo de *retropropagação de erro (error backpropagation*). Este algoritmo é baseado na regra de *aprendizagem por correção de erro.*

O PMC consiste em dois passos através das diferentes camadas da rede: um passo para frente, que chamamos de propagação, e um passo para trás, a retropropagação. No passo para frente, os vetores de entradas são aplicados aos neurônios da rede diretamente ligados e seus resultados se propagam através da rede até os últimos neurônios das últimas camadas, sabendo que outras saídas dos neurônios farão objetos de entradas de outros neurônios ligados, assim sucessivamente. Durante o passo de propagação, os pesos sinápticos da rede são todos fixos. Durante o passo para trás, por outro lado, os pesos sinápticos são todos ajustados de acordo com a regra de correção de erro. Eq. (2.5). Especificamente, a saída desejada é subtraída pela saída calculada para produzir um sinal de erro ($\boldsymbol{\varepsilon}$). Este sinal de erro é então propagado para trás através da rede, contra a direção das conexões sinápticas. [1]. Vindo daí o nome de "*retropropagação"*. O processo de aprendizagem realizado com o algoritmo é chamado de *aprendizagem por retropropagação.*

Um perceptron de multicamadas tem três características principais:

1. Cada neurônio da rede tem uma *função de ativação não-linear*. E a não linearidade é suave, quer dizer, diferençável em qualquer ponto. Ao contrário do perceptron de única camada onde tínhamos uma limitação abrupta que assumia somente o valor de 0 ou 1. Uma forma normalmente utilizada de não-linearidade que satisfaz esta exigência é uma não-linearidade sigmoide definida pela função [1]:

$$\boldsymbol{Y_j(t) = \frac{1}{1+e^{-U_j(t)}}} \tag{2.6}$$

   Em que U_j é a soma ponderada de todas as entradas sinápticas do neurônio *j* acrescentadas do bias, e Y_j é a saída do neurônio.

2. A rede tem uma ou mais camadas de neurônios intermediárias que não são parte da entrada ou da saída da rede. Estes neurônios intermediários capacitam a rede a aprender tarefas complexas extraindo progressivamente as características mais significativas dos vetores de entradas.

3. A rede apresenta um alto grau de conectividade, determinado pelas sinapses da rede.

É através da combinação destas características, juntamente com a habilidade de aprender da experiência através de treinamento, que o perceptron de multicamadas deriva seu poder. A presença de uma forma distribuída de não-linearidade e a alta conectividade da rede tornam difícil a análise teórica de um perceptron de multicamadas. E também a utilização de neurônios intermediários torna o processo de aprendizagem mais difícil de ser visualizado.
A Fig. 2.5 mostra a arquitetura de uma rede de perceptrons multicamadas com três camadas de neurônios intermediários. A propagação e a correção de erro no sentido contrário.

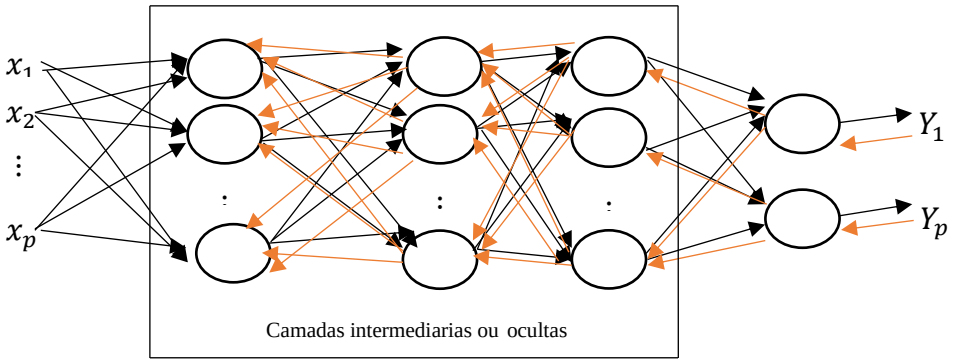

Figura 2.5. Arquitetura funcional de PMC com ilustração de dois fluxos

1. Sinais funcionais. Um sinal funcional é um sinal de entrada que se propaga de neurônio por neurônio através da rede até a saída. Ele realiza uma função útil na saída da rede e, a cada neurônio que passa, o sinal é calculado como uma função de suas entradas e pesos associados, aplicados àquele neurônio.
2. Sinais de Erro ($\boldsymbol{\varepsilon}$). Um sinal de erro começa nos neurônios da camada de saída e se propaga de camada por camada até a entrada da rede.

Os neurônios de saída constituem a camada de saída da rede. Os neurônios restantes constituem as camadas intermediarias da rede. Cada neurônio da camada intermediaria ou da saída de um perceptron multicamadas é projetado para realizar dois cálculos :

1. Em cada neurônio terá o cálculo do sinal funcional, que é calculado como uma função não-linear do sinal de entrada e dos pesos sinápticos associados diretamente com aquele neurônio.

2. A primeira derivada do erro que chamamos de gradiente da superfície de erro, será em relação aos pesos conectados às entradas de um neurônio, que é importante para a retropropagação através da rede.

O gradiente do algoritmo de retropropagação é muito importante, pois ele serve para atualização dos pesos. Essa derivada precisa de um pouco de atenção:
Notação:

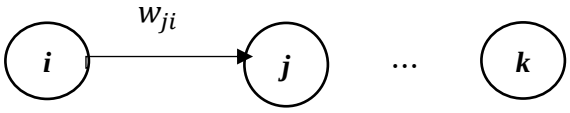

Figura 2.5. Ilustração das conexões de neurônios de diferentes camadas

- Os índices ***i***, ***j*** e ***k*** se referem a neurônios diferentes na rede; com os sinais se propagando através da rede da esquerda para a direita, o neurônio ***j*** se encontra em uma camada à direita do neurônio ***i,*** e o neurônio ***k*** se encontra em uma camada à direita do neurônio ***j***, quando o neurônio ***j*** é uma unidade oculta como mostrado na Fig. 2.5**.**
- O símbolo $\boldsymbol{\varrho}$(t) representa à soma instantânea dos erros quadráticos ou energia do erro na iteração t. A média de $\boldsymbol{\varrho}$(t) sobre todos os valores de t.
- O símbolo $\boldsymbol{\varepsilon}$(t) representa o sinal de erro na saída do neurônio ***k***
- O símbolo d_k(t) representa a resposta desejada para o neurônio ***k*** e é usada para calcular ε_k(t)
- O símbolo Y_k(t) representa à resposta do sinal funcional que aparece na saída de cada neurônio em cada iteração.
- O símbolo w_{ji}(t) representa o peso sináptico conectando a saída do neurônio ***i*** a entrada do neurônio ***j*** como ilustrado na Fig.2.5
- A soma ponderada de todas as entradas sinápticas acrescida do *bias* em relação ao neurônio ***j*** na iteração t é representado por U_j(t); constitui o sinal aplicado à função de ativação associada com o neurônio ***j***.
- A função de ativação, que descreve a relação funcional de entrada-saída da não-linearidade associada ao neurônio ***k***, é representada por φ_k(**.**).
- O bias aplicado ao neurônio j é representado por Θ_j; muitas vezes será por $w_{j0} = \Theta_j$.
- O sinal de entrada ou vetor de entrada é representado por x_i(t).
- O parâmetro de passo de aprendizagem é representado por η.

2.7. ALGORITMO DE *BACKPROPAGATION*

O sinal de erro é calculado somente nos últimos neurônios da camada de saída porque somente a camada de saída que tem a saída desejada d_k(t). Então o erro será definido como:

$$\varepsilon_k(\mathrm{t}) = d_k(\mathrm{t}) - Y_k(\mathrm{t}) \tag{2.7}$$

A energia do erro para o neurônio ***k*** como $\frac{1}{2}\varepsilon_k{}^2(t)$.

Depois de ter calculado os erros de todos os neurônios da camada de saída, vamos somar esses erros. Podemos assim escrever:

$$\varrho(t) = \frac{1}{2}\sum \varepsilon_k{}^2(t) \tag{2.8}$$

Seja *N* o número total de vetores de entrada contidos no conjunto de treinamento. A energia média do erro quadrado é obtida dividindo $\boldsymbol{\varrho}(\mathrm{t})$ por *N*.

$$\varrho_{med} = \frac{1}{N}\sum_{n=1}^{N} \varrho(t) \tag{2.9}$$

Os ajustes dos pesos são realizados de acordo com os respectivos erros calculados para cada vetor de entrada apresentado à rede. A média aritmética destas alterações individuais de peso sobre o conjunto de treinamento é, portanto, uma estimativa da alteração real que resultaria da modificação dos pesos baseada na minimização da função de custo ϱ_{med} sobre o conjunto de treinamento inteiro.

Vejamos então a Fig. 2.6, que mostra como calcular os sinais funcionais produzidos por uma camada de neurônios para outros.

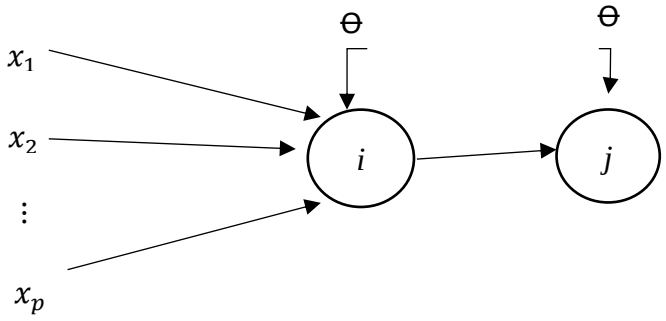

Figura 2.6. Demonstração de como calcular o sinal funcional.

Para o neurônio ***i*** teremos como sinal funcional:

$$U_i(t) = \sum_{i=1}^{p} w_{ip}(t)x_p(t) \tag{2.10}$$

Para o neurônio ***j*** teremos como sinal funcional :

$$U_j(t) = \sum_{j=1}^{n} w_{ij}(t)Y_i(t) \tag{2.11}$$

Assim, o sinal funcional que aparece na saída do neurônio j é

$$Y_j = \varphi_j(U_j(t)) \tag{2.12}$$

O algoritmo de backpropagation aplica uma correção $\Delta w_{ji}(t)$ aos pesos sinápticos $w_{ji}(t)$ que resulta da derivada parcial $\frac{\partial \varrho(t)}{\partial w_{ji}\,(\mathrm{t})}$.

Aplicando a regra da cadeia, podemos expressar este gradiente como:

$$\frac{\partial \varrho(t)}{\partial w_{ji}\,(\mathrm{t})} = \frac{\partial \varrho(t)}{\partial \varepsilon_j\,(\mathrm{t})}\frac{\partial \varepsilon(t)_j}{\partial Y_j\,(\mathrm{t})}\frac{\partial Y(t)_j}{\partial U_j\,(\mathrm{t})}\frac{\partial U(t)_j}{\partial w_{ji}\,(\mathrm{t})} \tag{2.13}$$

Fazendo a derivada de cada termo, teremos:

$$\frac{\partial \varrho(t)}{\partial \varepsilon_j\,(\mathrm{t})} = \varepsilon_j(t) \tag{2.14}$$

$$\frac{\partial \varepsilon(t)}{\partial Y_j\,(\mathrm{t})} = -1 \tag{2.15}$$

$$\frac{\partial Y(t)}{\partial U_j\,(\mathrm{t})} = \varphi_j'(U_j(t)) \tag{2.16}$$

Onde o uso da apóstrofe no lado direito do símbolo da função de ativação significa a diferenciação em relação ao argumento. Por exemplo, no caso da função sigmoide ficará:

$$\varphi_j'\left(U_j(t)\right) = \frac{-e^{-U_j(t)}}{\left(1+e^{-U_j(t)}\right)^2} \tag{2.17}$$

$$\frac{\partial U(t)}{\partial w_{ji}\,(\mathrm{t})} = Y_i \tag{2.18}$$

$$\frac{\partial \varrho(t)}{\partial w_{ji}\,(\mathrm{t})} = -\,\varepsilon_j(t)\,\varphi_j'\left(U_j(t)\right)Y_i(t) \tag{2.19}$$

A correção $\Delta w_{ji}(t)$ aplicado a $w_{ji}(t)$ é definida pela *regra delta:*

$$\Delta w_{ji}(t) = -\eta\frac{\boldsymbol{\partial \varrho(t)}}{\boldsymbol{\partial w_{ji}\,(\mathrm{t})}} \tag{2.20}$$

Para facilitar nossas contas, vamos ter uma variável chamada gradiente local:

$$\delta_j(t) = \varepsilon_j(t)\,\varphi_j'\left(U_j(t)\right) \tag{2.21}$$

O gradiente local aponta para as modificações necessárias nos pesos sinápticos. Um fator chave envolvido no cálculo do ajuste de peso $\Delta w_{ji}(t)$ é o sinal de erro que aparece na saída dos últimos neurônios da camada de saída. Percebemos que o cálculo de erro é somente nos neurônios da camada de saída porque lá se encontra a saída desejada. A questão é, entretanto, saber como penalizar ou recompensar os neurônios ocultos pela sua parcela de responsabilidade. Ele é resolvido de forma elegante retro propagando os sinais de erro através da rede.

2.8. IMPLEMENTAÇÃO DO ALGORITMO *BACKPROPAGATION.*

Na aplicação do algoritmo de backpropagation, distinguem-se dois passos distintos de computação. O primeiro passo é conhecido como passo para frente, ou propagação, e o seguinte como passo para trás, ou backpropagation.

No passo para frente, os pesos sinápticos se mantêm inalterados, até porque ainda não temos o sinal de erro. Lembre-se que o sinal de erro só aparece na camada de saída e os sinais funcionais da rede são calculados individualmente, neurônio por neurônio. Quando chegar nos últimos neurônios da camada de saída terá uma saída final $y_k(t)$ que será comparada com a saída desejada $d_k(t)$, obtendo-se o sinal de erro $\varepsilon_k(t)$ para o k-ésimo neurônio de saída. Assim a fase de propagação começa na primeira camada

intermediária, com a presença do vetor de entrada, e termina na camada de saída calculando o sinal de erro de cada neurônio desta .

O passo de backpropagation, por outro lado, começa na camada de saída passando-se os sinais de erro para a esquerda através da rede, camada por camada, e recursivamente calculando o gradiente local de cada neurônio. Este processo recursivo permite que os pesos sinápticos sofram modificações de acordo com a regra delta. Para um neurônio localizado na camada de saída, o gradiente local é simplesmente igual ao sinal de erro daquele neurônio multiplicado pela primeira derivada da sua não linearidade. Assim, utilizamos a formula das atualizações dos pesos de todas as conexões que alimentam a camada de saída.

Como falamos aqui, só explicamos brevemente sobre redes neurais artificiais porque fizemos uso deles como ferramenta para resolver um problema mais à frente. Há vários tipos de função de ativação, mas aqui usaremos a função sigmoide. Também vale ressaltar que nosso modo de aprendizagem é por lote, quer dizer, o ajuste dos pesos é realizado após a apresentação de todos os exemplos de treinamento que constituem uma época. Lembrando que falamos de erro quadrático médio na Eq. 2.8.

Com essas informações, podemos descrever como implementar o algoritmo de backpropagation. A atualização sequencial dos pesos é o método preferido para a implementação em tempo de execução on-line do algoritmo de backpropagation. Para este modo de operação, o algoritmo circula através da amostra de treinamento. Amostra de treinamento é a entrada e saída correspondente $\{x(t),\ d(t)\}_{n=1}^{N}$. O algoritmo funciona da seguinte maneira :

1. Primeiro passo é inicialização de variáveis. Não se preocupe de valores iniciais dos pesos pois serão modificados nas alterações. Inicializa com inteligência para não leva muito tempo na aprendizagem, senão a divergência será muito demorada e o sistema acaba sendo enjoado.
2. Apresente as entradas à rede. A cada entrada apresentada faz-se computação para frente, calculando o valor de ativação que será aplicado na função de ativação em cada neurônio para produzir a saída. Calcule os campos locais induzidos e os sinais funcionais da rede prosseguindo para frente através da rede, camada por camada. O campo local induzido U_j^c(t) para o neurônio *j* na camada *c* é

$$U_j^{(c)}\ (t) = \sum_{i=0}^{m} w_{ji}^c\ (t) Y_i^{c-1} \tag{2.22}$$

Em que $Y_i^{c-1}(t)$ é o sinal função de saída do neurônio *i* na camada anterior *c-1*, na iteração *t*, e w_{ji}^c (t) é o peso sináptico do neurônio *j* na camada *c*, que é alimentado pelo neurônio i da camada *c-1*. Assumindo-se o uso de função sigmoide, o sinal de saída do neurônio *j* na camada *c* é

$$Y_j^c = \varphi_j(U_j(t))$$

3. Quando se chega à última camada, então podemos calcular o erro do sinal, fazendo a diferença entre a saída desejada com a saída calculada.

$$\varepsilon_k(t) = d_k(t) - Y_k(t) \tag{2.23}$$

i, *j* e *k* são índices dos neurônios. Por exemplo na Eq. (2.23), *k* se refere ao neurônio da camada de saída.

4. Chegamos no final, agora é a computação para trás, backpropagation. Primeiro o cálculo do gradiente local da rede, como já foi dito.

$$\delta_j^c = \begin{cases} \varepsilon_j^c \varphi_j'(U_j^c(t) & para\ o\ neurônio\ j\ da\ última\ camada\ de\ saída \\ \varphi_j'(U_j^c(t) \sum_k \delta_k^{c+1}(t) w_{kj}^{c+1}(t) & para\ o\ neurônio\ j\ na\ camada\ intermediária \end{cases} \tag{2.24}$$

Ajuste os pesos sinápticos da rede na camada *c* de acordo com a regra delta generalizada:

$$w_{ji}^c(t+1) = w_{ji}^c(t) + \eta \delta_j^c(t) Y_i^{c-1}(t) \tag{2.25}$$

Em que η representa o passo de aprendizagem.

5. Repita os itens 3 e 4 para frente e para trás até satisfazer a condição de treinamento.

Conforme se pode observar ao longo deste capítulo, inúmeros são os parâmetros possiveis de serem ajustados, com o intuito de se melhorar o desempenho do processo de treinamento do PMC [22].

2.9. RESUMO

Finalmente, considerando também outros aspectos importantes envolvendo as fases de treinamento e operação do PMC, apresentam-se as seguintes notas práticas [22]:

- Atenta ao fato de que o aumento de neurônios e de camadas do PMC não implica diretamente em melhoria de seu potencial de generalização;
- Iniciar todas as matrizes de pesos com valores aleatórios pequenos;
- Impor uma quantidade máxima de épocas como critério adicional de parada do algoritmo de treinamento do PMC, pois é uma estratégia simples e eficiente para cessar o treinamento quando a precisão especificada se torna inalcançável;
- Normalizar os valores de entrada e saída das amostras visando evitar as regiões de saturação das funções de ativação.
- Adotar preferencialmente a tangente hiperbólica como função de ativação para os neurônios das camadas escondidas, pois a sua caraterística de anti-simetria (função ímpar) contribui para melhorar o processo de convergência da rede durante o respeito treinamento;
- Assumir sempre os dados dos subconjuntos de testes para avaliação do potencial de generalização.

No próximo capítulo vamos falar de uma rede neural multicamada, chamada função de base radial, que apresenta técnicas e arquiteturas diferentes do PMC.

CAPÍTULO 3: FUNÇÕES DE BASE RADIAL

3.1. INTRODUÇÃO

As funções de base radial podem resolver todos os problemas que PMC solucionam, inclusive aqueles que envolvem aproximação de funções e classificação de padrões. Diferentemente das redes PMC, que são compostas de várias camadas intermediárias, porém, a estrutura típica da RBF é composta por apenas uma camada intermediaria, na qual as funções de ativação são do tipo gaussiana. A estrutura dela tem uma camada de entrada, seguida de uma camada intermediária, que tem um ou mais neurônios, por fim uma camada de saída. Veja a Fig. 3.1.

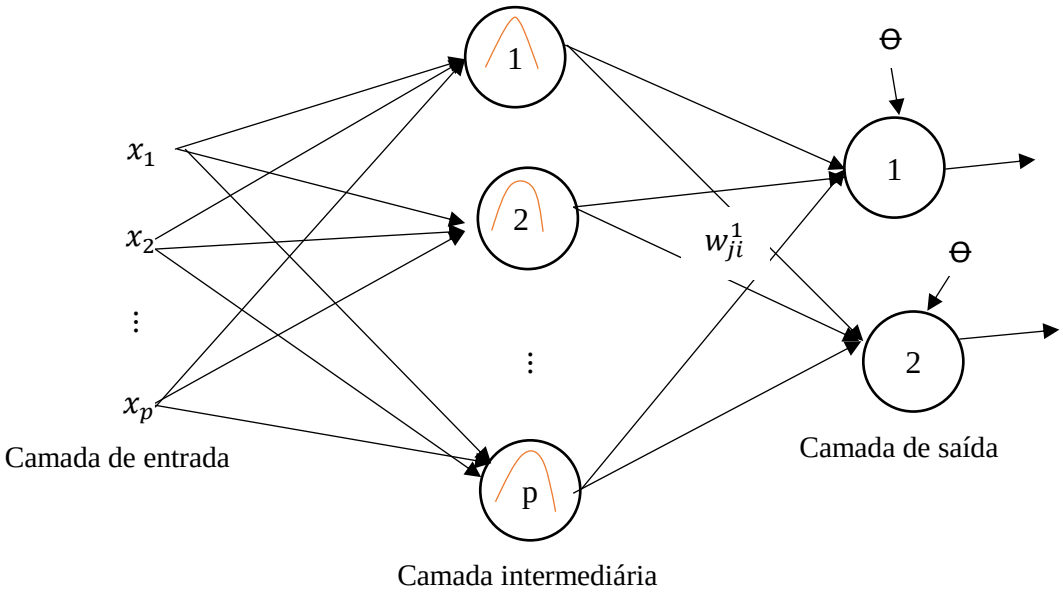

Figura 3.1 Arquitetura de um RBF

Uma das principais particularidades da rede RBF está na estratégia de treinamento utilizada para os ajustes dos seus pesos que será detalhado na próxima seção. Olhando a Fig.3.1, pode se observar que a camada escondida usa a função de ativação não linear que é a função gaussiana e a camada de saída por sua vez usa uma função linear, que é uma particularidade da rede RBF.

A rede RBF pertence também à arquitetura feedforward de camadas múltiplas, cujo treinamento é efetivado de forma supervisionada a partir da Fig. (3.1), verifica-se que o fluxo de informações na sua estrutura se inicia na camada de entrada, percorre então a respectiva camada intermediaria, neurônio por neurônio com a função de ativação gaussiana, finalizando seguidamente na camada neural de saída com uma função de ativação linear.

3.2. PROCESSO DE TREINAMENTO DE REDES RBF

O processo de treinamento das redes RBF não foge muito da rede PMC, em que cada uma de suas entradas $\{x_i\}$, representando os sinais vindo da aplicação, será então propagada pela referida camada intermediária em direção à camada de saída.

Entretanto, diferentemente do PMC, a estratégia de treinamento da RBF é constituída de dois estágios bem distintos entre si. O primeiro estágio tem como objetivo transformar uma função não linear dos dados de entrada para uma função linear. Ele faz uma distância euclidiana entre os dados de entradas e o centro por cada neurônio da camada intermediária. Já o segundo estágio, vinculados aos ajustes dos pesos dos neurônios da camada de saída, utiliza um critério de aprendizagem similar aquele usado na última camada do PMC, ou seja, a regra delta generalizada [22].

3.2.1. **Camada intermediária (estágio 1)**

Conforme mostrado na Fig. 3.1, os neurônios da camada intermediária da RBF são constituídos de funções de ativação do tipo gaussiana. A função gaussiana é representada como o seguinte:

$$\varphi(U) = e^{-\frac{U}{2\sigma^2}} \tag{3.1}$$

$\boldsymbol{\sigma}$ é o desvio padrão, mostra o quão larga está a função de ativação em relação ao centro.
No primeiro estágio calculamos essa distância:

$$U(t) = \sum_{i=1}^{m} \left|\left|x_i(t) - k\right|\right|^2 \tag{3.2}$$

Em que ||.|| é a distância euclidiana.

Essa conta é feita para cada neurônio da camada escondida

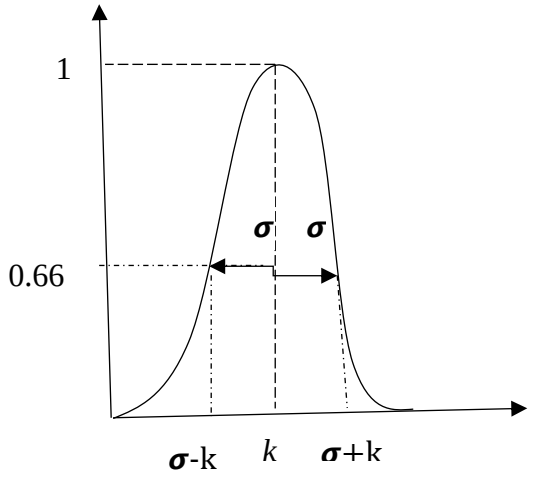

Figura 3.2. Função de ativação do RBF do tipo gaussiana

Assim, o principal objetivo dos neurônios da camada intermediária é posicionar os centros de suas gaussianas de forma mais apropriada possível. Um dos métodos muito bem utilizados para esta finalidade é denominado de *k-means* (k-médias), um dos propósitos é posicionar os centros de k-gaussianas em regiões onde os padrões de entrada tenderão a se agrupar.

3.2.2. **Camada de saída (estágio 2)**

Aqui vamos transformar a função não linear para uma função linear

$$Y_j(t) = \sum_{i=1}^{m} w_i \varphi_i(U(t)) \tag{3.3}$$

Vale ressaltar que os pesos sinápticos existem somente na camada de saída, sendo isso, a atualização dos pesos que ligam a camada intermediária e a camada de saída. A escolha dos centros aqui não importa muito, mas o número de centros deve ser menor que o número de amostras. As atualizações dos pesos utilizam a regra delta implementada no algoritmo backpropagation, que é a diferença entre a saída desejada e a saída real ou calculada. Comparando com o algoritmo backpropagation, a rede RBF é bastante simples para implementar. [22].

3.3. ALGORITMO DE APRENDIZAGEM DE REDES RBF

A sequência de passos para a aprendizagem do algoritmo:

1. Identificar as amostras, ou seja, as entras e suas respectivas saídas,$\{(x_i, d_i)\}$.
2. Especificar o passo de aprendizagem (η) e a precisão de treinamento;
3. Iniciar os pesos;
4. Calcular as distâncias euclidianas entre as entras e os centros que se encontra nos neurônios da camada intermediária;
5. Aplicar essas distâncias na função de ativação (gaussiana)
6. Associar as saídas dos neurônios da camada intermediárias com respectivos pesos para obter Y_j;
7. Calcular os erros ($d_j - Y_j$) correspondentes para cada amostra;
8. Calcular o erro médio;
9. Atualizar os pesos usando a regra delta;
10. Repetir o ponto (7) até que a condição da precisão requerida seja satisfeita.

3.4. APLICABILIDADES DAS REDES RBF

As redes RBF têm sido largamente utilizadas em problemas que envolvem aproximação de funções (identificação de sistema) e classificação de padrões, embora haja ainda diversos tipos de outras aplicações em que tenham sido implementadas com sucesso.

3.4.1. Classificação de Padrão

A rede RBF e PMC têm possibilidades classificar os padrões de entradas. A RBF consegue classificar um conjunto de amostras que não sejam linearmente separáveis. Basicamente, um mapeamento não-linear é usado para transformar um problema de classificação não–linearmente separável em um problema linearmente separável. Faremos um exemplo na próxima sessão para esclarecer melhor essa aplicação.

3.4.2. Aproximação de função

O motivo de incentivar a aplicabilidade das redes RBF como problema de interpolação é devido ao seu uso nesse projeto. A rede RBF é projetada para realizar um mapeamento não-linear de espaço de entrada para o espaço intermediário, seguindo de um mapeamento linear do espaço intermediário para o espaço de saída. O problema geral de aproximação de uma função multidimensional (várias entradas e uma saída), se trata de maneira seguinte: seja $f(x)$ uma função continua de $R^d \rightarrow R$, e $F(w, x)$, uma função que depende somente de $w \in R^P$ *e* x, onde d é a dimensão no espaço de funções.

Nesse contexto, o problema de aprendizagem se trata de determinar os parâmetros de uma função de aproximação tendo um conjunto finito l de pontos (x_i, d_i), ou seja, achar um hiperplano passando o máximo possível pelos pontos dados. Essa fase de treinamento constitui a otimização de procedimento de ajuste do hiperplano, baseando nos pontos dos dados conhecidos apresentados à rede na forma amostras entradas e saídas. Mas ao mesmo tempo, estamos procurando obter uma solução generalizada, quer dizer, uma estimação correta da função sobre uma região onde os pontos não são dados [2].

Isso nos leva diretamente ao problema de interpolação. A interpolação, em si, se define como: Dado um conjunto de N pontos diferentes ((x_i, d_i) | i = 1,2,3 ... N), encontre uma função F: $R^N \rightarrow R$ que satisfaz a condição de interpolação:

$$F(x_i) = d_i, i = 1,2, \dots N \tag{3.4}$$

A função *F* é obrigada a passar por todos os pontos dos dados de treinamentos. Lembrando quando tem amostras, separa um conjunto para treinamento da rede e outro conjunto para o teste da rede, é exatamente aqui precisaremos dessa noção.

A função de base radial (RBF) consiste em escolher uma função F que tem a seguinte forma:

$$F(x) = \sum_{i=1}^{N} w_i \varphi(||x - k_i||, \sigma_i) = \sum_{i=1}^{N} w_i h_i(x) \tag{3.5}$$

Onde ($\varphi\left(||x - k_i||\right)$ representa a função de base radial, e ||.|| representa a distância euclidiana. Igualando as duas equações lineares Eq. (3.4) e Eq. (5.5) para coeficientes desconhecidos da expansão $\{w_i\}$:

$$w_i h_i = d_i \tag{3.6}$$
$$w = h^{-1} d \tag{3.7}$$

Onde h é uma matriz formada pela função de base radial e w é uma matriz pesos.

3.5. DIFERENÇAS ENTRE RBF E PMC

- As duas redes são feedforward
- As duas são redes aproximadores universais
- RBF tem única camada escondida e o PMC tem uma ou mais camada escondida
- A camada es condida do RBF é não linear, e a camada de saída é linear
- A camada escondida e a camada de saída do PMC são não lineares
- O argumento da função de ativação de cada neurônio da camada escondida na rede RBF é calculado fazendo a distância euclidiana entre o vector de entrada e o centro do neurônio
- O argumento da função de ativação de cada neurônio da camada escondida do PMC é calculado fazendo a soma ponderado de todos vetores de entrada com seus respectivos pesos sinápticos.

3.6. RESUMO

Contrastando com os perceptrons de múltiplas camadas treinados com o algoritmo de retropropagação, o projeto de redes RBF usa somente a regra delta para atualizar seus pesos. Escolhe um centro adequado para cada neurônio da camada intermediária, e calcular as distâncias euclidianas com os sinais de entradas.
Concluindo, as funções de base radial foram inicialmente introduzidas nas soluções do problema de interpolação multivariada real.
No próximo capítulo mostra o uso da rede radial para resolver o problema de uma equalização cega.

CAPÍTULO 4: EQUALIZAÇÃO CEGA

4.1. INTRODUÇÃO

Um transmissor transmite uma informação através um meio de comunicação. Essa informação é transmitida até o receptor e, sendo corrompida pelo ruído introduzido no percurso ou pelas as interferências devido ao canal de comunicação. A pergunta é, como fazer para recuperar essa informação? Ao longo desse capítulo, vamos mostrar que é possível recuperar essa informação sem precisar do sinal piloto. Esse tipo de situação está presente no mundo real de comunicação e em tantas outras áreas, onde a informação enviada não chega inteiramente como enviada.

Geralmente, nos sistemas de comunicação, o canal é conhecido ou tem alguns traços para permitir a sua identificação. No entanto, aqui é diferente, não conhecemos o canal ou não temos nenhum traço que possa ajudar na identificação, a partir disso vem o nome de identificação do sistema cego: Equalização cega.

Aqui neste trabalho tratamos de uma abordagem de um equalizador cego baseado na rede neural artificial RBF. A RBF é usada como preditor para achar os coeficientes da saída do filtro inverso. Temos dois casos, o primeiro é um meio sem ruído e o segundo com ruído. Cada caso tem uma abordagem diferente em relação às técnicas de recuperação do sinal. Para um meio sem ruído usamos o LS (*least square*) e para o meio com ruído usamos o ILS (*improved least square*). Na seção seguinte explicaremos com detalhe cada uma dessas técnicas [27].

4.2. MODELO DO SISTEMA

Considere o sistema mostrado na Fig. 4.1, $c(t)$ representa o sinal gerado ou sinal de entrada. Podemos representar o sistema a ser identificado pelo modelo auto regressivo (AR). Auto regressão é um modelo matemático que permite descrever o comportamento futuro de um sistema em relação às suas ações anteriores. Escolhemos esse modelo devido a suas características e por ser muito utilizado nos sistemas de aplicações como a comunicação móvel e radio indoor, nesse caso a saída do sistema fica assim:

$$x(t) = -\boldsymbol{h^T x}(t-1) + c(t) \tag{4.1}$$

Esta é a representação matemática de um modelo autoregressivo, (AR). A saída $x(t)$ depende de um conjunto de resultados anteriores, $\boldsymbol{x}(t-1) = [x(t-1), x(t-2), x(t-3), \dots, x(t-p)]^T$, e p é a ordem do modelo de autoregressão, que deverá ser um número inteiro. Essa ordem é conhecida a priori. A notação $[.]^T$ representa uma matriz transposta. Depois disso, o sinal $x(t)$ é transmitido. O receptor final recebe o sinal $y(t)$ [27].

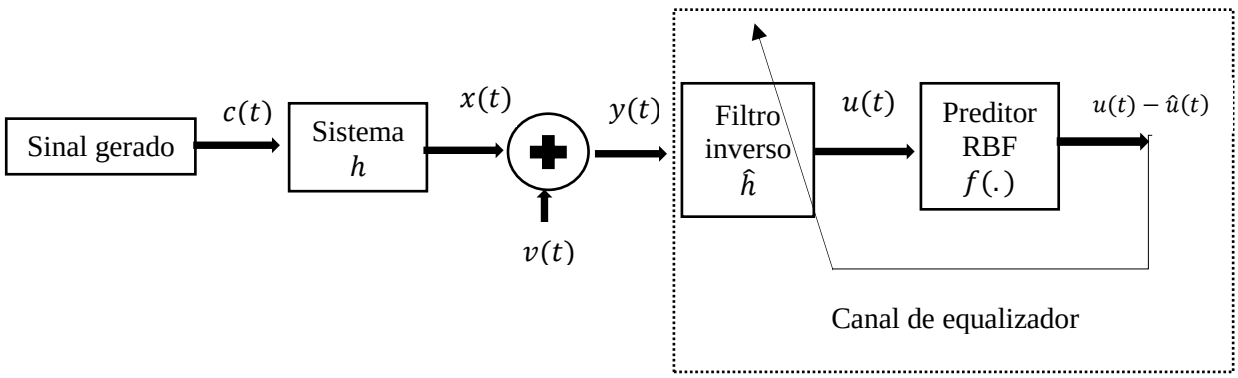

Figura 4.1. Diagrama de blocos do sistema de identificação cega [27].

O objetivo da equalização cega é justamente estimar os coeficientes h se-baseando somente no sinal $y(t)$.

$$y(t) = x(t) + v(t) \tag{4.2}$$

Em que $v(t)$ é o ruído Gaussiano branco, com média 0 e a variância σ^2 .

A abordagem da equalização cega é baseada sobre a habilidade do preditor de rede neural artificial, RBF. A ideia é concentrada sobre um conceito chamado *minimum nonlinear predict error (MNPE)* [27]. Assumimos que o sinal de entrada deve ser não linear, esta é a primeira condição. Então podemos dizer que:
$c(t) = f(\boldsymbol{c}(t-1))$
onde $\boldsymbol{c}(t-1) = [c(t-1), c(t-2), c(t-3), \dots, c(t-d)]^T$ e d é a dimensão do sinal de entrada e as letras em negrito são vetores [27].

O preditor RBF está colocado no sistema já treinado e testado, chamado treinamento *off-line*.. O treinamento e teste da RBF é feito com as amostras do sinal gerado $\boldsymbol{c}(t)$. O RBF terá várias entradas com uma única saída. Lembrando do capítulo anterior, onde foi comentado que o RBF também poder ser usado para aproximar uma função não linear, portanto, usaremos o RBF para aproximar a função não linear do sinal $c(t)$.

$$c(t) = g\big(\boldsymbol{c}(t-1)\big) + \boldsymbol{\zeta} \tag{4.3}$$

$$g(\boldsymbol{c}(t-1)) = \sum_{j=1}^{M} w_j e^{\left(-\frac{\left\|c(t-1)-k_j\right\|^2}{2\sigma_j^2}\right)} \tag{4.4}$$

Em que $g(.)$ é o RBF, $\boldsymbol{\zeta}$ é o erro de aproximação, e k é o centro.

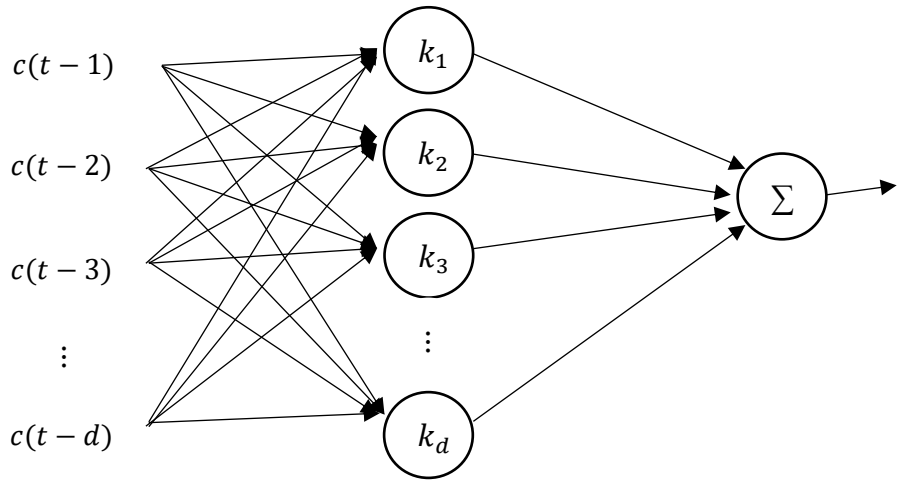

Figura 4.2. Rede RBF com *d*-entradas e uma saída.

A partir das Eq. (4.3) e Eq. (4.4) o sinal de entrada já pode ser substituído pela função de RBF. A Fig. 4.2, mostra o funcionamento do RBF, com várias entradas e uma única saída. Vamos mostrar sua ligação com o filtro inverso. O $y(t)$ chega no filtro inverso e a saída será representada por $u(t)$. Como o modelo é autoregressivo, então a saída do filtro inverso pode ser escrita desta forma [27]:

$$u(t) = y(t) + \widehat{\boldsymbol{h}}^{\boldsymbol{T}}(\boldsymbol{y}(t-1)) \tag{4.5}$$

Se conseguimos obter perfeitamente a saída do filtro inverso, podemos dizer que é igual ao nosso sinal de entrada que foi aproximada pela função de base radial (RBF). Isso será nossa hipótese, que diremos

($u(t) = c(t)$) e, baseando-se nisso, percebemos que só temos uma incógnita, que são os coeficientes $\hat{h}$. A solução dos coeficientes do filtro inverso é feita sobre a regra do mínimo quadrado.

$$\min_{\hat{h}^T} \phi_{LS}\left(\hat{h}\right) = \min_{\hat{h}} E[\left(u(t) - f\left(u(t-1)\right)\right)^2] \tag{4.6}$$

Com as Eq. (4.3) e (4.4), a equação (4.7) muda, pois, a função $f(.)$ é aproximada pela rede neural RBF

$$\min_{\hat{h}^T} \phi_{LSRBF}\left(\hat{h}\right) = \min_{\hat{h}} E[\left(u(t) - g\left(u(t-1)\right)\right)^2] \tag{4.7}$$

Em que $g\left(u(t-1)\right) = \hat{u}$(t), é a saída da RBF

Agora que temos a diferença entre a saída do filtro inverso e a saída do RBF, podemos por meio de gradiente decrescente achar os coeficientes do filtro inverso. Quando aplicamos o método do MNPE (*minimum nonlinear preditor error*), teremos o seguinte [27]:

$$\hat{h}(t+1) = \hat{h}(t) - \frac{\eta}{2}\frac{\partial \phi_{LSRBF}\left(\hat{h}\right)}{\partial \hat{h}(t)} \tag{4.8}$$

$$\frac{\partial \phi_{LSRBF}\left(\hat{h}\right)}{\partial \hat{h}(t)} = 2(u(t) - \hat{u}(t))\boldsymbol{y}(t-1) \tag{4.9}$$

Em que, $\left(u(t) - \hat{u}(t)\right) = erro(\boldsymbol{\varepsilon})$.

Essa técnica de achar os coeficientes $\hat{h}$ do filtro inverso é baseado na regra delta, também visto nos capítulos anteriores. Este método trata-se da minimização da regressão do erro quadrático médio da saída do filtro inverso $u(t)$ com seus respetivos vetores de atrasos $u(t-1)$.

O segundo caso do algoritmo ILS RBF é quando $u(t)$ está corrompido com os ruídos e a equação (4.8) não será mais adequada para recuperar o sinal de entrada $c(t)$, mostra-se sua ineficiência nos resultados. Neste caso, usaremos outra equação que vem da técnica ILS-RBF-MNPE [28], onde teremos:

$$\phi_{ILSRBF}\left(\hat{h}\right) = E[\frac{r^2(t)}{1 + \nabla_x g(\boldsymbol{u}(t-1))\nabla_x^T g(\boldsymbol{u}(t-1))\hat{h}(t)}] \tag{4.10}$$

Em que $r(t) = y(t) - \widehat{h^T}(t)g(\boldsymbol{u}(t-1))$,
e $\nabla_x g = \frac{\partial g}{\partial \boldsymbol{u}(t-1)}$, derivada parcial da função RBF.

Se, $\Psi(t) = 1 + \nabla_x g\left(\boldsymbol{u}(t-1)\right)\nabla_x^T g\left(\boldsymbol{u}(t-1)\right)\hat{h}(t)$, então a equação (4.10) fica:

$$\phi_{ILSRBF}\left(\hat{h}\right) = E[\frac{r^2(t)}{\Psi(t)}] \tag{4.11}$$

Faz-se a deriva primeira da função do RBF, teremos:

$$\nabla_x g = \sum_{j=1}^{M} -w_j(u(t-1) - k_j)e^{\left(-\frac{\left(u(t-1) - k_j\right)^2}{2\sigma^2}\right)} \tag{4.12}$$

Seguindo a mesma derivada feita na Eq. (4.8), a técnica do gradiente procura minimizar o erro do quadrático medio. Neste caso o ILS-MNPE fica:

$$\hat{h}(t+1) = \hat{h}(t) - \frac{\eta}{2}\frac{\partial \phi_{ILSRBF}\left(\hat{h}\right)}{\partial \hat{h}(t)} \tag{4.13}$$

Em que, $\frac{\partial \phi_{ILSRBF}(\hat{h})}{\partial \hat{h}(t)} = -2\frac{r(t)g(u(t-1))\Psi(t)\hat{h}(t)+r^2(t)\nabla_x g(\boldsymbol{u}(t-1))\nabla_x^T g(\boldsymbol{u}(t-1))\hat{h}(t)}{\Psi^2(t)}$ (4.14)

4.3. IMPLEMENTAÇÃO DO SISTEMA

A implementação foi dividida em duas partes. A primeira parte é um sistema sem ruído e segunda parte um sistema com ruído no canal de comunicação.

1º. Implementação do sistema sem ruído (LS RBF-MNPE):

1. Definir o sinal de entrada;
2. Descrever a saída do sistema em relação ao sinal de entrada;
3. Calcular o sinal de saída $y(t)$;
4. Com as amostras definidas em função do sinal de entrada, treinar o RBF e testá-lo também;
5. Descrever a saída do filtro inverso;
6. Aplicar os respectivos atrasos de $u(t-d)$ no RBF para obter $\hat{u}$(t);
7. Calcular o erro ($\boldsymbol{\varepsilon}$) conforme a Eq. (4.7);
8. Atualizar os coeficientes do filtro inverso conforme a Eq. (4.8);
9. Repetir os itens 5 a 8 até que o erro convirja.

2º. Implementação do sistema com ruídos (ILS RBF-MNPE):

1. Definir o sinal de entrada;
2. Descrever a saída do sistema em relação ao sinal de entrada;
3. Calcular o sinal $y(t)$;
4. Achar os centros dos neurônios e os pesos do RBF para usar na derivada;
5. Com as amostras definidas em função do sinal de entrada, treinar o RBF e testá-lo também;
6. Descrever a saída do filtro inverso;
7. Aplicar os respectivos atrasos de $u(t-d)$ no RBF para obter $\hat{u}$(t);
8. Calcular a derivada primeira da função RBF conforme a Eq. (4.12);
9. Calcular o erro ($\boldsymbol{\varepsilon}$) que chamamos de $r(t)$ aqui;
10. Atualizar os coeficientes do filtro inverso conforme a Eq. (4.14);
11. Repetir os itens 6 a 11 até que o erro convirja.

4.4. FUNÇÃO DE MAPA LOGÍSTICO

Um mapa logístico é uma função matemática que associa um dado número x_n a um outro número x_{n+1} através da equação (4.15). [29]

$$x_{n+1} = \lambda\, x_n(1 - x_n) \quad (4.15)$$

Onde **λ** é um parâmetro de controle.

O modelo de mapa logístico foi descrito pelo biólogo Robert May em 1976 como um modelo populacional para insetos, com x_n sendo o número de indivíduos na enésima geração e **λ** como uma taxa de crescimento da população. A equação do mapa logístico é estudada por vários motivos e por possuir muitas vantagens [29]:

I. Ele é acessível. Os estudantes de ensino médio podem estudar este modelo dispondo apenas de uma calculadora de mão ou um pequeno computador.

II. Ele é exemplar. É um simples exemplo que ilustra muitas noções fundamentais de dinâmica não-linear, apresentando equilíbrio, periodicidade, caos, bifurcação e fractais.
III. Apresenta uma matemática cheia de vida e de riqueza. Muitos aspectos da equação logística ainda não são rigorosamente entendidas e são estudadas por alguns dos melhores matemáticos.
IV. Ele é relevante para a ciência. Previsões derivadas do mapa logístico foram verificadas em experimentos em fluidos com fraca turbulência, oscilações de reações químicas, circuitos elétricos não-lineares e uma variedade de outros sistemas.

Obtemos a série temporal $\{x_n, n = 0,1,2,3, ...\}$ do mapa logística dado pela equação 4.13 iterando recursivamente o mapa a partir de um ponto inicial x_0, condição inicial, que depende da escolha de um valor para o parâmetro **λ**, mantido fixo durante a iteração da série, mas que pode ser alterado para que possamos obter outra série.

O gráfico do mapa logístico é uma parábola com concavidade voltada para baixo. O vértice desta parábola está diretamente ligado ao valor do parâmetro **λ**. Para construir o mapa logístico é escolhido um valor inicial qualquer $x_0 \in [0\ 1]$; por este ponto traça-se uma reta perpendicular ao eixo x_n que cruza a parábola; o ponto de encontro entre a reta e a parábola corresponde a uma valor x_1 no eixo x_{n+1}. Encontrado este valor devemos passar uma reta horizontal exatamente por cima desse ponto e que encontre a reta $y = x$ que, por possuir propriedades simétricas, nos auxilia a encontrar exatamente o mesmo valor x_1, mas no eixo x_n. [29].

O máximo da parábola, mostrado na Figura 4.3 é encontrado fazendo-se a derivada da função logística, onde obtemos:

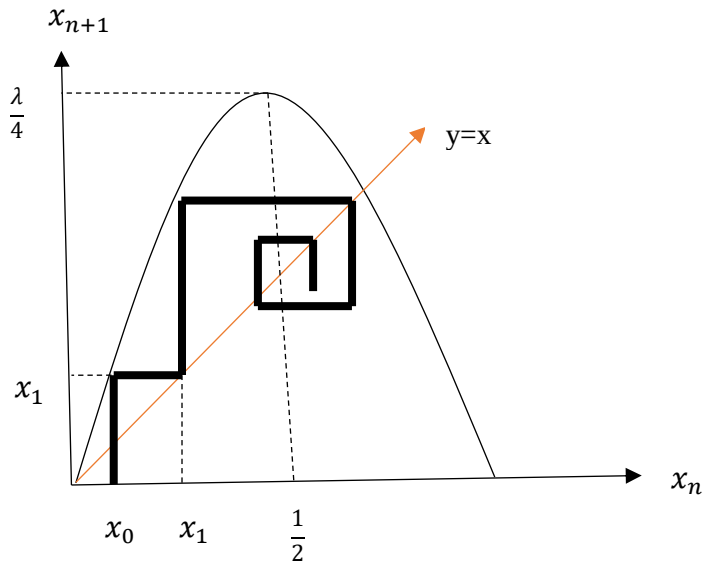

Figura 4.3. Demonstra o funcionamento de um mapa logístico com diferentes valores [29].

Para encontrar o ponto x, que é máximo da parábola, devemos igualar a derivada da função a zero. Sabendo que, quando a derivada for zero, teremos o seu máximo já que esta equação nos dá uma parábola com concavidade voltada para baixo e, sendo assim, ela só terá um máximo. Isolando a variável, temos:

$$f'(x) = \lambda - 2\lambda x \qquad (4.16)$$
$$\lambda - 2\lambda x = 0 \quad => \quad x = \frac{1}{2}$$

Agora que sabemos o valor de x, onde teremos o máximo da parábola, fazemos a $f(\frac{1}{2})$ da equação logística e encontramos, finalmente, o máximo:

$$f\left(\frac{1}{2}\right) = \lambda\left(\frac{1}{2}\right) - \lambda\left(\frac{1}{2}\right)^2 = \frac{\lambda}{4} \tag{4.17}$$

Na subseção 4.4 fizemos um estudo do mapa logístico e falamos que suas caraterísticas dependem diretamente do valor de λ. Nosso objetivo agora é introduzir a noção do comportamento caótico. Vamos mostrar que tal comportamento caótico aparece em sistemas nos quais conhecemos a equação que descreve a sua evolução temporal.

No diagrama da Fig. 4.3 observamos que o mapa logístico vai sofrendo uma serie de bifurcações e aproximadamente a partir de λ = 3.6 é praticamente impossível, apenas observando o gráfico, distinguir que tipos de órbitas aparecem. No entanto, como veremos, essas órbitas se tornam caóticas. Vamos analisar duas propriedades importantes que caracterizam o comportamento caótico, aperiodicidade e sensibilidade às condições iniciais.

- **Aperiodicidade:** sabemos que numa órbita periódica, após certa quantidade de iterações, um determinado ponto da órbita volta ao ponto de partida, ou seja, a órbita apresenta um padrão e podemos fazer previsões futuras sobre o modelo. No caso de aperiodicidade, temos um sistema que não atende nenhum ponto, nem a um ciclo periódico e também não diverge par o infinito. As iterações do mapa ficam andando aleatoriamente dentro de um determinado intervalo. Essas órbitas são chamadas de órbitas caóticas, pois não apresentam nenhum tipo de padrão ou período. Em tais órbitas é muito difícil fazer precisões futuras sobre o sistema devido à ausência de um comportamento padrão. [33].

- **Sensibilidade às condições iniciais:** vimos anteriormente que para determinados valores do parâmetro λ o sistema pode passar de um regime estável para um regime caótico. Para exemplificar uma das características mais importantes do caos, a sensibilidade as condições iniciais vão analisar a órbitas do mapa logístico com λ = 4 para duas condições iniciais próximas. Podemos observar na Fig. 4.4, que duas condições iniciais, mesmo sendo bem próximas, deram origem a duas órbitas diferentes após 2500 amostras. Uma figura gerou o sinal esperado, portanto outra gerou somente quatros pontos espalhados na superfície. Em sistemas que não são caóticos duas condições iniciais próximas geram órbitas que continuam próximas. Por outro lado, no regime caótico, uma pequena mudança nas condições iniciais gera uma grande diferença nas órbitas a longo prazo. Mesmo que houvesse uma forma de determinar o comportamento de um sistema caótico a partir de sua condição inicial, isso seria impraticável, pois os instrumentos utilizados para fazer medições são limitados, tornando impossível conhecermos com exatidão o estado inicial de um sistema. Sendo assim, o desconhecimento do estado inicial com precisão acarreta na impossibilidade de fazermos previsões futuras sobre o sistema. Novamente recaímos sobre o problema o problema de prever sistemas caóticos. A ideia de que pequenas causas geram grandes efeitos é uma maneira informal de enunciar a sensibilidade nas condições iniciais, que é uma questão central quando se fala de caos. [10]

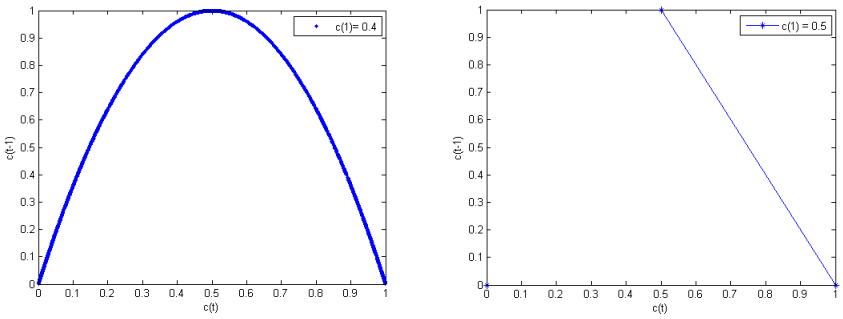

Figura 4.4. Dois mapas logísticos com valores iniciais próximos, mas com reações bem diferentes

4.5. PROBLEMAS E RESULTADOS

Seja uma função de mapa logístico caótico com $\boldsymbol{\lambda}= 4$.

$$c(t) = \lambda c(t-1)(1-c(t-1)) \tag{4.17}$$

O ponto inicial do mapa logístico foi 0.8. Nosso modelo AR é de dimensão 2 com coeficientes $h = [0.195, -0.95]$. O sinal $c(t)$ foi gerado com 2500 amostras. Veja a Fig. (4.5).

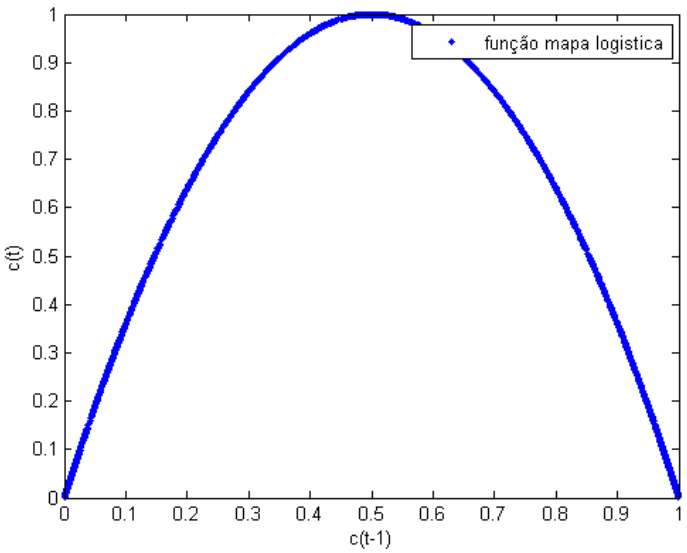

Figura 4.5. Sinal de entrada, mapa logístico caótico.

A Fig. 4.5 mostra o sinal gerado pelo *c(t)*, este é o sinal de entrada. Escolhemos um sinal representado por mapa logístico caótico para mostrar o funcionamento e a importância de se usar as redes neurais artificiais para resolver problema da equalização cega. Já na figura seguinte Fig. 4.6, mostra o sinal de entrada no domínio do tempo, em que, o sinal é aperiódico. Nenhum ponto se repete em nenhuma iteração, a cada ponto a função apresenta características diferentes.

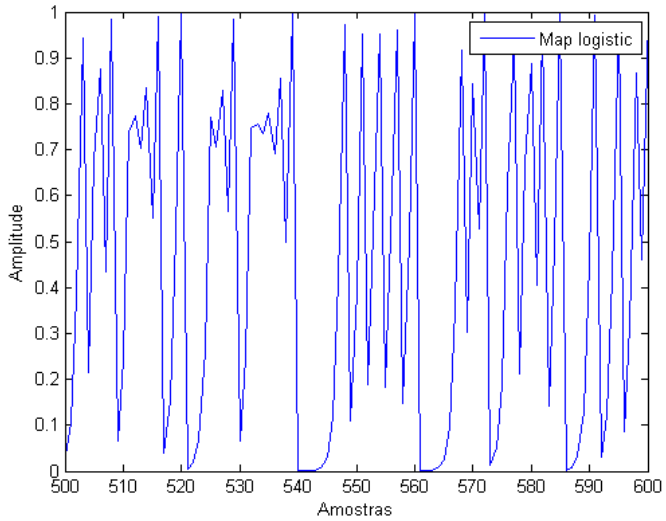

Figura 4.6. Estimação do sinal de entrada no domínio do tempo.

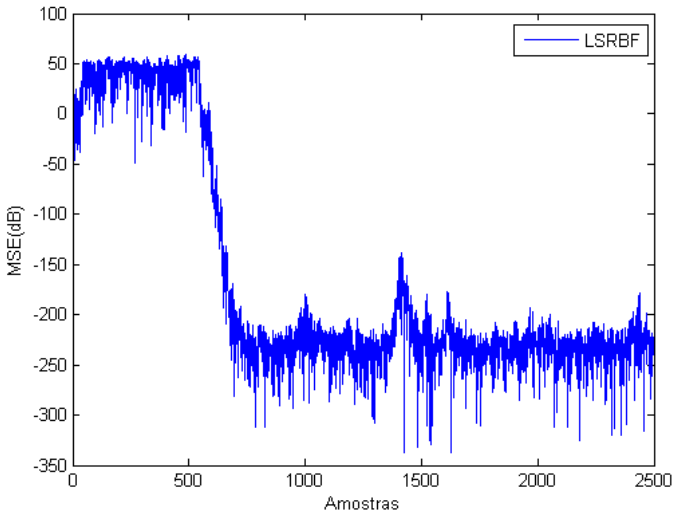

Figura 4.7. Erro quadrático médio estimado entre a saída do sistema inverso e a saída do preditor, RBF.

Na Fig. 4.7, mostra a evolução do erro quadrático médio do algoritmo LSRBF,

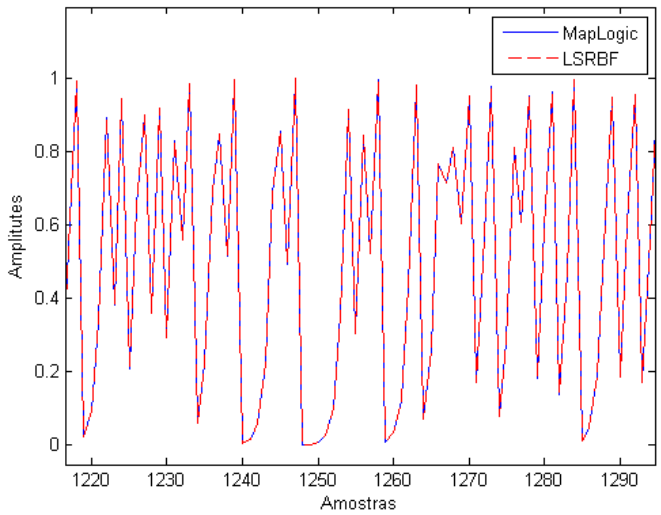

Figura 4.8. Funcionamento do algoritmo LS RBF MNPE.

A Fig. 4.8, depois de aplicamos o algoritmo de LSRBF, como deve-se observar o sinal de entrada está representado pela cor azul, e o sinal aproximado pela cor vermelha. Um perfeito funcionamento do sistema, o sinal vermelho segue ou faz uma perfeita superposição sobre a cor azul, que nos mostra justamente o bom funcionamento do algoritmo e a grande importância de se usar as redes neurais artificiais. Depois da simulação, a saída do filtro inverso nos dá esses coeficientes $\widehat{h} = [0.195 \quad 0.95]$ que são iguais aos coeficientes do sistema.

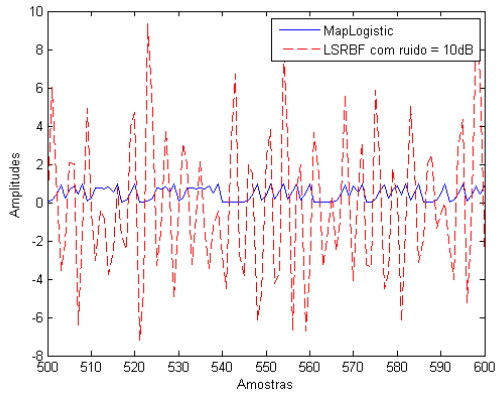

Figura 4.9 (a)

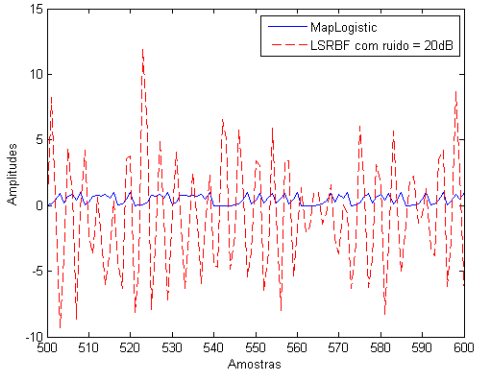

Figura 4.9 (b)

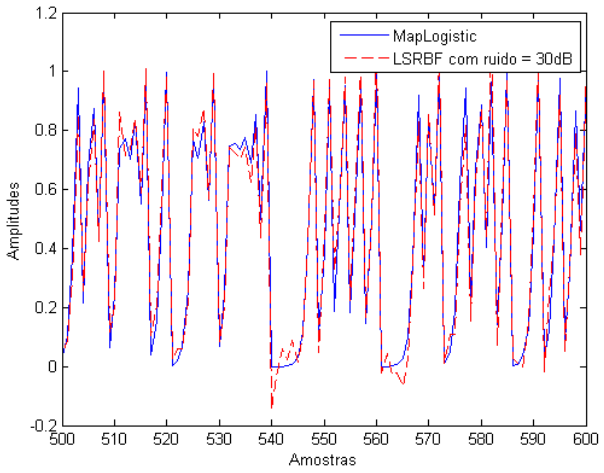

Figura 4.9 (c). Figuras demonstrativos do algoritmo LS-RBF-MNPE no meio ruidoso.

Observe-se as Fig. 4.9. (a, b, e c) o algoritmo LS-RBF funciona no meio ruidoso, mas não apresentou os resultados esperados como na Fig. 4.8. Logo em seguindo, mostra a implementação o algoritmo ILS-RBF.

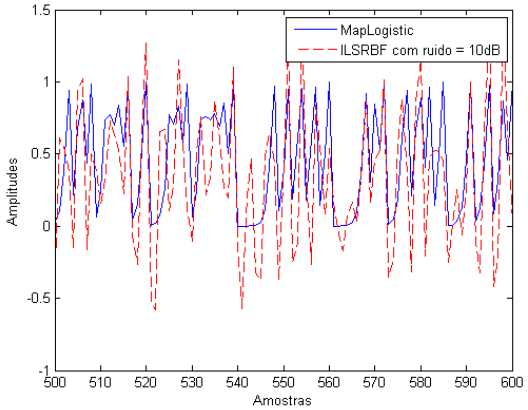

Figura 4.10 (a)

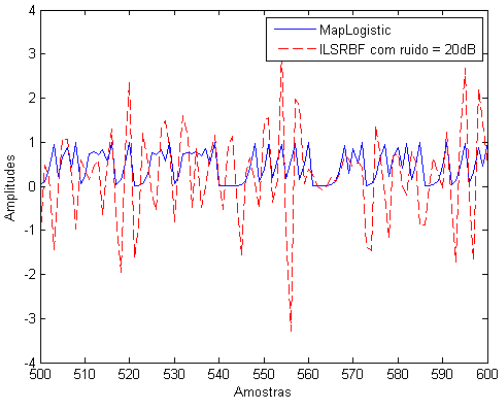

Figura 4.10 (b)

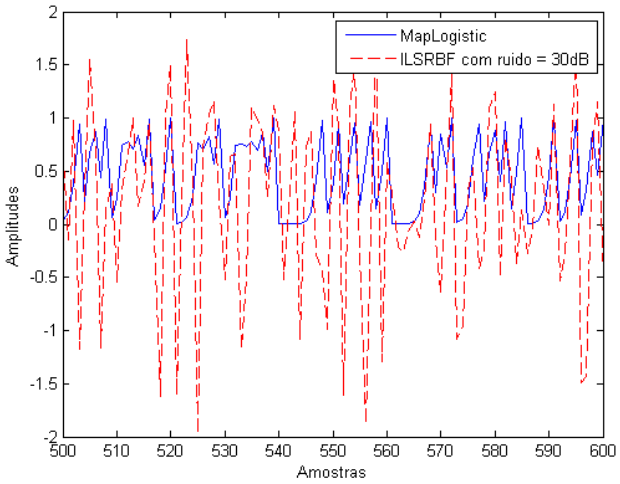

Figura 4.10. (c). Figuras demonstrativos do algoritmo ILS RBF MNPE no meio ruídos.

Olhando as Fig. 4.10 (a, b e c) vê-se o funcionamento do algoritmo ILS-RBF no meio ruidoso. O sinal da cor vermelha consegue acompanhar o sinal azul com uma pequena diferença que se justifica pela presença dos ruídos. Para comparar os dois algoritmos no meio ruidoso, foi variado o SNR de 0 a 40, e comparando os erros quadráticos médios.

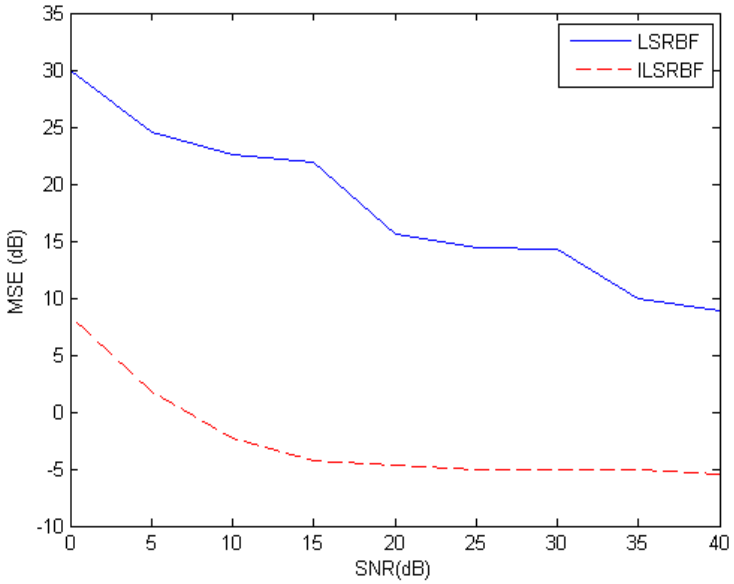

Figura (4.11). Comparação do MSE dos dois algoritmos LSRBF e ILSRBF

Deve – se observar na Fig. 4.11 o bom funcionamento do algoritmo ILSRBF no meio ruidoso. O algoritmo tem uma melhor performance do que LSRBF. Os resultados dos algoritmos LSRBF e ILSRBF foram obtidos em uma única iteração com 2500 amostras. O MSE foi calculado fazendo a diferença entre o sinal geral e o sinal calculado na saída do filtro inverso, depois foi feito a média das 2500 amostras, por fim convertido em decibel.

4.6. RESUMO.

Este capítulo tratou-se de equalização cega, usando a rede neural artificial do tipo RBF como preditor para identificar um sistema não-linear num ambiente sem ruído e com ruído. Usando a função de base radial para prever a saída do filtro inverso, é mostrado aqui que o sistema desconhecido pode ser identificado minimizando o erro na saída do filtro inverso. Para mostrar que o sistema funciona, foi implementado dois algoritmos LSRBF e ILSRBF cujos os resultados encontram-se nas Fig. 4.8 e Fig. 4.9.

CAPÍTULO 5: CONCLUSÃO E TRABALHO FUTURO

Este trabalho tratou da proposta de um equalizador cego utilizando uma rede neural do tipo RBF. A rede RBF usou 20 neurônios na camada escondida, com um MSE de 2.6507910^{-12}. A rede RBF teve 2001 amostras para o treinamento e 499 para teste. A taxa de aprendizagem dos algoritmos LSRBF e ILSRBF foi de 0,05.

O capítulo 1, tratou do histórico e criação das redes neurais artificiais. A ideia começou a partir das observações dos neurônios biológicos e do cérebro humano. A capacidade e a eficiência que um ser humano tem de aprender as coisas simultaneamente, então isso levaram os primeiros pesquisadores para criar os neurônios artificiais, daí vem, as redes neurais artificias.

O capítulo 2, fala das estruturas e algoritmos de aprendizagem dos neurônios artificiais, principalmente do algoritmo de *Backpropagation*. As diferentes camadas de uma rede neural artificial foram vistas, e os funcionamentos de cada uma. As funções de ativação que são implementadas em cada neurônio e suas importâncias. Mostra-se o uso da regra delta para atualização dos pesos sinápticos.

O capítulo 3, na sua vez, fala da rede RBF que é um modelo de rede neural multicamadas, portanto, com uma particularidade. Ela tem uma única camada escondida e aplica funções de ativações diferentes. Uma para camada escondida que é a função gaussiana e outra para a camada de saída, que é uma combinação linear. Por isso, diz-se, que a RBF transforma uma função não linear para uma função linear. As atualizações dos pesos respeitam a mesma regra delta, que é baseada na diferença das saídas desejada com a saída real.

O capítulo 4, por fim, trata da equalização cega usando a rede neural artificial do tipo RBF para identificar os coeficientes do filtro inverso. Usa-se um sinal de entrada to tipo mapa logístico, um sistema caótico, aperiódico e sensível nas condições iniciais, justamente para mostrar o funcionamento dos algoritmos LS-RBF-MNPE e ILS-RBF-MNPE. O equalizador identifica um sistema não linear, minimizando o erro, entre a saída do filtro inverso e a saída do RBF. Com a regra delta, pode-se recuperar o sinal, identificando os coeficientes do filtro inverso.

Como trabalhos futuros, pode-se verificar o desempenho do equalizador cego estudado em problemas de detecção de radar, equalização de sinais de voz e enlaces de comunicação para equalização do canal móvel variante no tempo, substituindo o modelo logístico por outras fontes de informação.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] HAYKIN, Simon. *Redes neurais: princípios e prática/ Simon Haykin*; trd. Paulo Martins Engl. - 2.ed. – Porto Alegre: Bookman, pp 902, 2001.

[2] Aspray, W., and A. Burks, 1986. Papers of John Von Neumann on computing and computer Theory Charles Babbage institute Reprint Series for The History of computing, Vol 12. Cambridge, MA : MIT Press.

[3] Rochester, N, J. Holland, L.H Haibt, and WL. Duda, 1956 " Tests on a cell assembly theory of the action of the brain, using a large digital computer, " IRE Transaction on information theory, vol. IT-2, pp 80-93.

[4] Uttley, A.M., 1956 " A theory of the mechanism of learning basead on the computation of conditional probabilities " Proceeding of the First internacional conference on cybernetics, Namur, Gauthier-Villars, Paris.

[5] Caianiellos, E.R., 1961, "Outline of a theory of thought-processes and thinking machines," journal of theoretical Biology, vol.1, pp- 204-235.

[6] Uttley, A. M, 1979, Information Transmission in the Nervous System, London : Academic Press

[7] Steinbuch, K. 1961, " Dielernmatrix," Kybernetic, Vol.1, pp 36-45.

[8] Anderson, J.A., 1972, "A simples neural network generating na interative memory," Matenatical Biosciences, Vol. 14, pp 197-220.

[9] Kohonen, T. 1972. " correlation matrix memories ," IEEE transaction on computers, Vol. C-21, pp 353-359.

[10] Nakano, K., 1972. "Association –a model of associative memory," IEEE transactions on Systems, Man, and Cybernetics, Vol. SMC – 2, pp 380 – 388.

[11] Rosembatt, F., 1958, " The Perceptron: A probabilistic model for information storage and organization in the brain, " psychological Review, Vol 65, pp 386 – 408.

[12] Rosemblatt, F., 1960b. " On the convergence of reinforcement procedures in simples perceptrons," cornell Aeronautical laboratory Report, VG-1196–G-4, Buffalo, NY.

[13] Novikoff, A.B.J., 1962. " On convergence proofs for perceptrons ," in proceeding of the symposium on the matematical theory of automata, pp. 615 – 622, Brooklyn, NY : Polytechnic institute of Brooklyn.

[14] Widrow, B., 1962 " Generalization and information storage in networks of adaline " neurons, in M.C. Yovitz, G.T. Jocobi, and G.D., Goldstein , eds., self-Organizing systems, pp. 435 – 461, washington, DC : spartan Books.

[15] Nilson, N.J., 1965. Learning Machines: Foundation of Trainable Pattern – classifying Systems, New York : MC Graw – Hill.

[16] Minsky. M. L and S.A Papert. 1969, perceptron, cambridge, M.A : MIT press.

[17] Minsky. M.L O.G. Selfridge, 1961. " Learning in random nets ," information theory, Fourth London symposium, London. Butterworks.

[18] Cowan, J.D., 1990 " Neural networks : the early days advance in neural information processing systems, " Vol.2, pp. 828 – 842, San Mateo, CA : Morgan Kaufmann.

[19] Von der Malsburg, C., 1973 " Self-orgaization of orientation sensitive cells in the stuate cortex, " Kybernetik, vol.14, pp. 85 – 100.

[20] Ramón y Cajál., 1911, Histologie du systems nerveux de l'homme et des vertebres, Paris : Maloine.

[21] Shepherd, G.M., and C. Kosh, 1990, " introduction to synaptic circuits," in the synaptic organization of the Brain, G. M. Shepherd, ed., pp – 3-31. New York : Oxford University Press.

[22] SILVA, Ivan Nunes da; SPATTI, Danila Hernane; FLAUZINO, Rogerio Andrade. *Redes Neurais Artificiais para Engenharia e ciências aplicadas, curso prático*. ARTILIBER, pp 399, 2010.

[23] Mc Culloch, WS., and. Pitts, 1943, " A logical calculus of the ideas immanent in nervous activity," Bulletin of Mathematical biophysics, vol. 5, pp. 115 – 133.

[24] M. Q. Zhang, M. Z. Rong, and K. Friedrich, in : " Handbook of organic inorganic Hydrid materials and Nanacomposites, volume 2: nanocomposites," H. S Nalwa Ed., American scientifics Publishers, stevenson Ranch, CA, 2003, p 113.

[25] Parsons, S. and Jones, G. 2000. Acoustic identification of twelve species of echolocating bat by discriminant function analysis and artificial neural networks. J Exp Biol., 203 : 2631 – 2656.

[26] https://blogdopetcivil.com/2013/07/05/redes-neurais-artificiais/, acessado dia 20/06/2017, às 09:21

[27] Xan Xie *and* Henry Leung, "*blind Equalization using a predictive Radial Basis Função Neural Network"* IEEE Trans. Neural Netw, vol. 16, pp 709- 720 NO. 3. May 2005.

[28] Zhiwen Zhu and Henry Leung, 'Adaptive Identification of Naonlinear Systems with Application to Chaotic Communications'. IEEE Trans on Circuits and Systems : Fundamental Theory and Applications, Vol. 47, pp 1072-1080. NO. 7, july 2000.

[29] Fernanda Jaiara Dellajustina, Estudo do mapa logístico, Udesc Joinville, pp 23, 20 de dezembro de 2012, 12:43h.

[30] Kavita Burse, R.N. Yadav, and S.C. Shrivastava, "*Tecnical correspondence channel Equalization using Neural Networks: A review".* IEEE Trans. On Sys, Man, And Cybernetics-Part C: Applications and Review, vol. 40, pp 352-357. No. 3, may 2010.

[31] João Luís Garcia Rosa, SCC-5809 – Capítulo 6, Redes de função de Base Radial, pp 71, Universidade de São Paulo, Instituto de ciências Matemáticas e de computação, Departamento de ciências de Computação, 2011.

[32] Emmanuel Vienne, '*Réseaux á fonctions de base radiales', Younès Bennani.* Apprentissage connexionniste, Lavoisier, pp. 105, 2006, I2C Hernès, HAL Id :hal-00085092, https:// hal.archives-ouvertes.fr/hal-00085092.

[33] Morais, Leonardo, Equações de Diferenças, Caos e Fractais. Dissertação de mestrado – Universidade Federal de Santa Catarina, Centro de Ciências Físicas e Matemáticas. Programa de Pós-Graduação. pp 117. Florianópolis, SC, 2014.