

**LIGA DE ENSINO DO RIO GRANDE DO NORTE
FACULDADE NATALENSE PARA O DESENVOLVIMENTO DO RIO GRANDE DO
NORTE**

**IMPLEMENTAÇÃO DE UM PROTÓTIPO DE REDE NEURAL RBF (FUNÇÃO DE
BASE RADIAL) UTILIZANDO JAVA PARA ANÁLISE DA VARIAÇÃO
ARQUITETURAL E COMPORTAMENTAL**

EMERSON SOUZA DE SENA

*Trabalho de graduação, Curso de Sistemas
de Informação da Faculdade Natalense para o
Desenvolvimento do Rio Grande do Norte
como requisito obrigatório para obtenção do
título de Bacharel em Sistemas de
Informação.*

*Orientador: Claudemir de Assis da Rocha
Jana, MA.*

**IMPLEMENTAÇÃO DE UM PROTÓTIPO DE REDE NEURAL RBF (FUNÇÃO DE
BASE RADIAL) UTILIZANDO JAVA PARA ANÁLISE DA VARIAÇÃO
ARQUITETURAL E COMPORTAMENTAL**

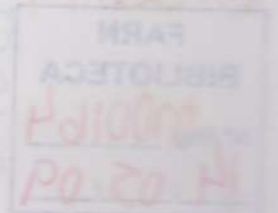
EMERSON SOUZA DE SENA

**IMPLEMENTAÇÃO DE UM PROTÓTIPO DE REDE NEURAL RBF (FUNÇÃO DE
BASE RADIAL) UTILIZANDO JAVA PARA ANÁLISE DA VARIAÇÃO
ARQUITETURAL E COMPORTAMENTAL**

Monografia apresentado, Curso de Sistemas de Informação da Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte como requisito obrigatório, para obtenção do título de Bacharel em sistemas de Informação.

Orientador: Gleydson de Azevedo Ferreira Lima, Msc.

NATAL / RN
2008



IMPLEMENTAÇÃO DE UM PROTÓTIPO DE REDE NEURAL RBF (FUNÇÃO DE
BASE RADIAL) UTILIZANDO JAVA PARA ANÁLISE DA VARIAÇÃO
ARQUITETURAL E COMPORTAMENTALCatalogação na Publicação – Biblioteca da FARN
Setor de Processos Técnicos

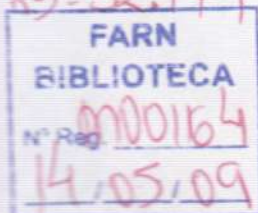
Sena, Emerson Souza de
Implementação de um protótipo de rede neural RBF (Função de Base
Radial) utilizando Java para análise da variação arquitetural e
comportamental / Emerson Souza de Sena. – Natal: FARN, 2008.
85f.

Orientador: Gleyson Lima (Msc.)
Monografia (Bacharelado em Sistemas de Informação) – Faculdade
Natalense para o Desenvolvimento do Rio Grande do Norte.

1. Inteligência Artificial – Monografia. 2. Redes Neurais Artificiais –
Monografia. 3. Operadores Lógicos – Monografia I.Título.

RN/FARN/BC

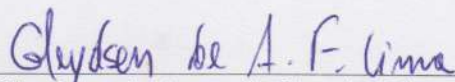
CDU 004.8



EMERSON SOUZA DE SENA

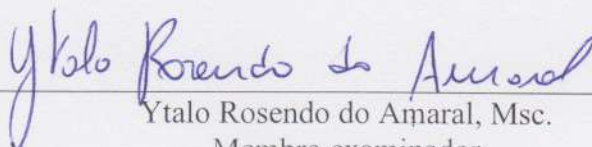
**IMPLEMENTAÇÃO DE UM PROTÓTIPO DE REDE NEURAL RBF (FUNÇÃO DE
BASE RADIAL) UTILIZANDO JAVA PARA ANÁLISE DA VARIAÇÃO
ARQUITETURAL E COMPORTAMENTAL**

Monografia apresentada ao curso de Bacharelado em Sistemas de Informação da Faculdade
Natalense para o Desenvolvimento do Rio Grande do Norte aprovada em ____ de ____
____ de ____, pela banca examinadora composta pelos seguintes membros.



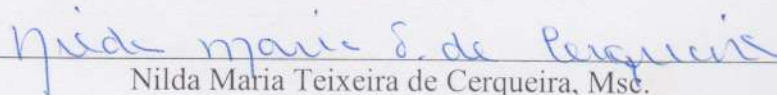
Gleydson de Azevedo Ferreira Lima, Msc.

Orientador



Ytalo Rosendo do Amaral, Msc.

Membro examinador



Nilda Maria Teixeira de Cerqueira, Msc.

Membro examinador

AGRADECIMENTOS

Agradeço a Deus pelo dom da vida e por todas as bênçãos que recebi durante os quatro anos da minha graduação, que me ajudaram a seguir nesta jornada.

Aos meus pais, Vilma Telma e Francisco Manoel, que sempre lutaram por mim, com amor, dedicação e por demais obrigações para eu alcançar meus objetivos. A vocês também, pela compreensão nos momentos em que não pude dar a devida atenção no decorrer da minha graduação.

A minha noiva, Gislene, por seu amor, compreensão, paciência, companheirismo e por estar sempre ao meu lado tanto nos momentos de tristeza como de alegrias. Agradeço também, pelo grande incentivo que deu à minha formação.

A meu sogro José Alves e minha sogra Marinalva Costa, por todo apoio, atenção e compreensão durante a minha vida caminhada.

A todos os meus familiares e amigos que apesar de não estar sempre perto, me incentivaram e me deram força para alcançar meus objetivos.

A todos os professores, especialmente ao meu orientador, por sua dedicação, e auxílio em minha formação.

Dedico este trabalho a toda
minha família pelo apoio e
compreensão nesta importante
fase da minha vida.

AGRADECIMENTOS

Agradeço a Deus pelo dom da vida e por todas as bênçãos que recebi durante os quatro anos da minha graduação, que me ajudaram a muito nesta jornada.

Aos meus pais, Vilma Telma e Francisco Manoel, que sempre torceram por minhas vitórias, pelo amor dedicado e por darem condições para eu alcançar meus objetivos. A meus irmãos, pela compreensão nos momentos em que não pude dar a devida atenção no decorrer da minha graduação.

A minha noiva Girlane, por seu amor, compreensão, paciência, companheirismo e por estar sempre ao meu lado tanto nos momentos de tristeza como de alegrias. Agradeço também, pelo grande incentivo que deu á minha formação.

A meu sogro José Alves e minha sogra Marinalva Costa, por todo apoio, atenção e compreensão dedicados a mim nesta caminhada.

A todos os meus familiares e amigos que apesar de não estar sempre perto, me incentivaram e me deram força para alcançar meus objetivos .

A todos os professores, especialmente ao meu orientador Gleydson Lima, por terem contribuído, e muito, em minha formação .

RESUMO

Este trabalho apresenta um estudo sobre a variação arquitetural e o comportamento de uma Rede Neural Artificial Função de Base Radial, observando o seu desempenho de acordo com a sua topologia. Para tanto implementou-se uma Rede Neural Artificial Função de Base Radial capaz de aprender e executar funcionalidades de três operações lógicas *XOR*, *AND* e *OR*, a fim de analisar alguns pontos específicos comportamentais da rede através de configurações e parametrizações, tendo como foco principal a quantidade de neurônios na camada escondida da Rede Neural Artificial Função de Base Radial.

PALAVRAS-CHAVE: Rede Neural Artificial, Inteligência Artificial, Função de Base Radial.

ABSTRACT

This paper presents a study on the architectural variation and behavior of an Artificial Neural Network Radial Base Function, noting its performance in accordance with its topology. For both put up an Artificial Neural Network Radial Base Function able to learn and perform functions of three logical operations XOR, AND and OR, to examine some specific points of the network by means of behavioral settings and parameters, with the main focus of the number of neurons in the hidden layer of Function Artificial Neural Network Radial Base Function.

KEY WORDS: Artificial Neural Network, Artificial Intelligence, Radial Base Function.

Figura 01 Neurônio Biológico.....	24
Figura 02 Representação do neurônio de McCulloch e Pitts.....	26
Figura 03 Camadas de uma Rede Neural Artificial.....	28
Figura 04 Estrutura de modelo de RNA's.....	29
Figura 05 Aprendizagem de uma Rede Neural Artificial.....	29
Figura 06 Neurônio artificial genérico.....	33
Figura 07 Função (semi-linear) e derivada em relação à entrada interna.....	34
Figura 08 Função (logística) e derivada em relação à entrada interna.....	34
Figura 09 Função (tangente hiperbólica) e sua derivada em relação à entrada interna.....	35
Figura 10 Esquema treinamento do <i>Perceptron</i>	37
Figura 11 Aprendizagem por correção de erro.....	38
Figura 12 Funções de Ativação.....	39
Figura 13 Modelo lógico genérico do neurônio.....	40
Figura 14 Aprendizagem auto-organizada.....	43
Figura 15 Rede Simétrica.....	45
Figura 16 Aprendizagem por reforço.....	46
Figura 17 Representação de competitividade entre neurônios.....	47
Figura 18 RNA neural recorrente.....	49
Figura 19 Arquitetura típica de uma RNA auto-organizável unidimensional.....	50
Figura 20 Arquitetura típica de uma RNA auto-organizável bidimensional.....	50
Figura 21 Possível distribuição de pesos para um SOM unidimensional.....	51
Figura 22 RNA <i>feedforward</i> com uma única camada.....	52
Figura 23 Matriz W (propagação).....	52
Figura 24 RNA <i>feedforward</i> de múltiplas camadas.....	53
Figura 25 Exemplo de superfície de erro.....	55
Figura 26 Processo competitivo: seleção do neurônio vencedor.....	56
Figura 27 Processo cooperativo: relações de vizinhança.....	57
Figura 28 Processo cooperativo: relações de vizinhança gaussiana.....	58
Figura 29 Processo adaptativo.....	59
Figura 30 Função decrescente.....	60
Figura 31 Função crescente.....	60
Figura 32 Rede neural artificial de base radial.....	62
Figura 33 Rede neural artificial de base radial com múltiplas saídas.....	62
Figura 34 Aproximação usando RNA RBF caso 1.....	67
Figura 35 Aproximação usando RNA RBF caso 2.....	67
Figura 36 Diagrama da aplicação.....	73
Figura 37 Classe RBFGui.....	74
Figura 38 Classe RBF.....	75
Figura 39 Classe NeurônioEscondido.....	76
Figura 40 Classe Treinamento.....	77
Figura 41 Classe Usar.....	78
Figura 42 Interface de Configuração.....	79
Figura 43 Interface de Treinamento.....	80
Figura 44 Interface de Utilização da RNA.....	80
Figura 45 Gráfico do número da média de iterações do operador XOR.....	81
Figura 46 Gráfico do número da média de iterações do operador AND.....	82
Figura 47 Gráfico do número da média de iterações do operador OR.....	82

SUMÁRIO

1 INTRODUÇÃO	11
1.1 O PROBLEMA	12
1.2 OBJETIVOS	13
1.2.1 Objetivo Geral	13
1.2.2 Objetivos Específicos	13
1.3 JUSTIFICATIVA	13
2 HISTÓRICO	14
3 INTELIGÊNCIA ARTIFICIAL	18
3.1 ABORDAGENS DA INTELIGÊNCIA ARTIFICIAL	19
3.2 CAMPOS DE APLICAÇÃO DA INTELIGÊNCIA ARTIFICIAL	20
3.3 MODELOS DE INTELIGÊNCIA ARTIFICIAL	21
3.4 FUNDAMENTAÇÃO BIOLÓGICA	23
4 REDES NEURAIS ARTIFICIAIS	24
4.1 O NEURÔNIO DE MCCULLOCH E PITTS	25
4.2 CARACTERÍSTICAS GERAIS DAS REDES NEURAIS	27
4.3 UNIDADES GENÉRICAS EM RNA'S	32
4.4 ALGUNS PARADIGMAS DE APRENDIZAGEM	35
4.5 APRENDIZAGEM COM SUPERVISÃO	36
4.5.1 Formalização Processo de Aprendizado com Supervisão	37
4.5.2 Tipos de Algoritmos de Aprendizado com Supervisão	39
4.5.2.1 O Perceptron	39
4.5.2.2 Perceptron Simples para Classificação de Padrões	39
4.5.2.3 Perceptron de Múltiplos Neurônios	41
4.5.2.4 Características Básicas do Treinamento do Perceptron	42
4.5.3 Aprendizagem sem Supervisão	43
4.6 TOPOLOGIAS DE REDES NEURAIS ARTIFICIAIS	44
4.6.1 Redes Neurais Simétricas	45
4.7 ALGUNS TIPOS DE REDES NEURAIS	45
4.7.1 Redes Neurais com Função Base Radial (RBF)	45
4.7.2 Redes de Aprendizagem por Reforço	46
4.7.3 RNA com Competitividade	47
4.7.4 Redes Neurais Recorrentes	48
4.7.5 RNA com Auto-Organização	49
4.7.5.1 Arquitetura de RNA com Auto-Organização	50
4.7.5.2 Mapas auto-organizáveis (Treinamento)	51
4.8 REDES <i>FEEDFORWARD</i>	52
4.8.1 Rede <i>Feedforward</i> com uma Única Camada	52
4.8.2 - Rede <i>Feedforward</i> de Múltiplas Camadas (MPL)	53
4.9 O ALGORITMO DE <i>BACKPROPAGATION</i>	54
4.10 PROCESSOS EM RNA'S	55
4.10.1 Processo Competitivo	55
4.10.2 Processo Cooperativo	56
4.10.3 Processo Adaptativo	58
4.11 FUNCIONAMENTO DE REDES NEURAIS ARTIFICIAIS RBF	59
4.11.1 Métodos de Treinamento	63
4.11.2 Capacidade de Aproximação Universal	64
4.11.3 O Método dos Quadrados Mínimos para Modelos Lineares	64

4.11.4 Obtenção da solução ótima	65
4.11.5 Aproximação usando RNA RBF	67
4.11.6 Técnicas para Determinação dos Centros e Dispersões	68
4.11.7 Seleção de Centros por Auto-Organização	69
5 METODOLOGIA	70
5.1 TIPO DE PESQUISA	70
5.1.1 Quanto aos Objetivos	70
5.1.2 Quanto aos Procedimentos	71
5.1.3 Quanto ao Objeto	71
6 RESULTADOS	72
6.1 DESCRIÇÃO DAS CLASSES DA APLICAÇÃO E INTERFACES	74
6.1.1 Classe RBFGui	74
6.1.2 Classe RBF	75
6.1.3 Classe Neurônio Escondido	76
6.1.4 Classe Treinamento	77
6.1.5 Classe Usar	78
6.2 INTERFACES DA APLICAÇÃO RNA RBF	79
6.3 TESTES E ANÁLISE COMPORTAMENTAL DA RNA RBF	80
7 CONCLUSÃO	83
REFERÊNCIAS	84

"Artificial" vem do latim *artificialis*, que significa algo não natural, mas é produzido pelo homem. Dessa forma, inteligência artificial é um tipo de inteligência produzida de forma não natural, para permitir que máquinas possam ter de alguma maneira que assemelhe a inteligência humana.

A IA tem se desenvolvido na busca por compreender a inteligência e replicar diversas funções do conhecimento com o objetivo primário de simulá-la. A inteligência artificial busca entender a mente humana e imitar seu comportamento, levantando questões como: Como ocorre o pensamento? Como o homem entende o contexto do mundo? Como é a memória, os sentidos e a linguagem quando se discute o conceito de inteligência? Como funciona a mente? Como a mente processa informações e toma decisões baseadas por uma rede de neurônios? Essas são algumas das perguntas que a IA precisa responder para simular o raciocínio humano e replicar aspectos da inteligência.

A IA sofreu influência de várias áreas da ciência, por isso existem várias ramificações da IA. Uma delas é a Neurociência, que se dedica ao estudo do sistema nervoso, em particular a estrutura. A forma como esse órgão habilita o pensamento é um dos grandes mistérios desta ciência. A partir do conhecimento dessa área, pode-se especificamente o estudo dos neurônios, se desenha o modelo de IA chamado de Redes Neurais Artificiais (RNA). As RNA's são estruturas que imitam fundamentalmente os mecanismos de processamento de dados humanos, e sua habilidade em tomar decisões não lineares com grande rapidez faz com que

1 INTRODUÇÃO

Atualmente ouve-se falar cada vez mais em robôs que podem interagir com o mundo real, aplicações que tem a capacidade de aprender e tomar decisões, entre outros. Grandes produções do mundo cinematográfico sintetizam isso de forma fictícia, onde máquinas inteligentes se comportam e realizam diversas tarefas interagindo aceitando e lidando com acontecimentos do mundo real, antes feitos apenas por seres humanos. Enfim, há uma área de pesquisa da ciência da computação em que o principal objetivo é buscar métodos ou dispositivos computacionais que simulem as faculdades humanas, estamos falando da Inteligência Artificial (IA).

A palavra “Inteligência” vem do latim *inter* (entre) e *legere* (escolher). Entende-se por “Inteligência” aquilo que permite ao ser humano escolher entre uma coisa e outra. Já a palavra “Artificial” vem do latim *artificiale*, que significa algo não natural, isto é produzido pelo homem. Desta forma, inteligência artificial é um tipo de inteligência produzida de forma não natural, para permitir que máquinas façam uso de alguma habilidade que simula a inteligência do homem.

A IA tem se destacado na busca por compreender a inteligência e englobar diversos campos do conhecimento com o objetivo prático de simulá-la. A inteligência artificial busca entender a mente humana e imitar seu comportamento, levantando questões como: Como ocorre o pensamento? Como o homem extrai o conhecimento do mundo? Como a memória, os sentidos e a linguagem ajudam no desenvolvimento da inteligência? Como surgem as idéias? Como a mente processa informações e tira conclusões decidindo por uma coisa ao invés de outra? Essas são algumas das perguntas que a IA precisa responder para simular o raciocínio humano e implementar aspectos de Inteligência.

A IA sofreu influência de várias áreas da ciência, por isso existem vários modelos de IA. Uma delas é a Neurociência, que se dedica ao estudo do sistema nervoso, em particular o cérebro. A forma como esse órgão habilita o pensamento é um dos grandes mistérios desta ciência. A partir do conhecimento dessa área mais especificamente o estudo dos neurônios, se deu início ao modelo de IA chamado de Redes Neurais Artificiais (RNA). As RNA's são sistemas não lineares fundamentados nos mecanismos de processamento do cérebro humano. A sua habilidade em formar mapeamentos não lineares com processamento paralelo e

distribuído, formado por unidades de processamento simples denominados neurônios, tem tornado as RNA's prósperas na modelagem de sistemas heurísticos complexos.

1.1 O PROBLEMA

Atualmente, pode-se observar que a contribuição da IA tem proporcionado avanços em áreas como: financeira, recursos humanos, marketing, medicina, engenharia e muitas outras, onde existem várias aplicações, na medida em que os requisitos de desempenho para os sistemas físicos tornam-se cada vez mais exigentes, a utilização de modelos linearizados tem trazido grande limitação operacional e, em algumas situações, pode até mesmo inviabilizar o projeto de alguns sistemas. A habilidade de lidar com dados ruidosos, incompletos ou imprecisos, e de prever sistemas não lineares, torna a aplicação de uma RNA bastante eficiente.

Estruturalmente, uma RNA consiste em um número de elementos interconectados chamados "neurônios", organizados em camadas que aprendem pela modificação da conexão, firmemente conectado as camadas.

Existe um grande interesse nos modelos de RNA's devido ao seu grande poder de aprendizagem, generalização e adaptação, que propicia a resolução problemas de grande complexidade. Os modelos de RNA's normalmente apresentam um conjunto de neurônios de entrada, por onde são fornecidas as informações para a RNA, um conjunto de neurônios de saída, que representam os sinais de saída da rede neural, e um conjunto de neurônios intermediários.

Visando o estudo e a análise da variação arquitetural e comportamental de uma Rede Neural Artificial RBF(Função de base Radial), implementaremos o modelo de RNA RBF, capaz de aprender e executar funcionalidades dos operadores lógicos (*XOR*, *AND* e *OR*), onde assim poderemos mensurar a variação e comportamento da rede em questão.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo do presente trabalho é desenvolver um protótipo de RNA RBF, capaz de aprender e executar funcionalidades dos operadores lógicos (*XOR*, *AND* e *OR*) para a análise do comportamento da rede em relação a variação arquitetural nos referidos operadores lógicos.

1.2.2 - Objetivos Específicos

- Realizar um estudo sobre redes neurais artificiais, analisando a aplicabilidade de técnicas no processo de reconhecimento de padrões;
- Realizar um estudo sobre os operadores lógicos que serão utilizados na implementação da RNA RBF;
- Analisar dados que serão utilizados na alimentação da RNA;
- Implementar a rede neural;
- Validar o modelo implementado;
- Treinar a RNA para cada operador lógico (*XOR*, *AND* e *OR*);
- Realizar testes e análise arquitetural e comportamental RNA.

1.3 JUSTIFICATIVA

Atualmente, pode-se observar que a contribuição da IA (Inteligência Artificial) tem proporcionado avanços em áreas como: financeira, recursos humanos, marketing, medicina, engenharia e muitas outras, onde existem várias aplicações. Dentro da IA, há diversos

paradigmas em relação à arquitetura computacional, dentre eles está o de RNA's (Redes Neurais Artificiais). As RNA's são sistemas não lineares fundamentados nos mecanismos de processamento do cérebro humano. A sua habilidade em formar mapeamentos não lineares com processamento paralelo e distribuído, formado por unidades de processamento simples denominados neurônios, tem tornado as RNA's prósperas na modelagem de sistemas heurísticos complexos. Implementou-se o modelo de RNA RBF(Função de base Radial), capaz de aprender funcionalidades dos operadores lógicos (*XOR*, *AND* e *OR*) para a análise do comportamento da rede em relação a variação arquitetural nos referidos operadores lógicos, com o intuito de obter informações peculiares que venham possivelmente melhorar a performance e eficácia, o que é de extrema importância quando se trata de sistemas heurísticos complexos.

2 HISTÓRICO

Hoje a IA encontra-se em um nível estrutural muito fundamentado, porém para que isso fosse possível, a contribuição de outras áreas, com idéias, pontos de vistas e técnicas, foi de extrema importância. Entre elas estão:

- **Filosofia** de 428 a.C até a atualidade, onde são estudados pontos como: As primeiras regras formais com conclusões válidas; como a mente (o intelecto) pode se desenvolver a partir de um cérebro físico; de onde vem o conhecimento; como o conhecimento conduz a ação. Muitos filósofos contribuíram através do tempo: Aristóteles, Leonardo da Vinci, Thomas Hobbes, entre outros (RUSSELL; NORVING, 2004).
- **Matemática** cerca do ano 800 a.C até a atualidade, onde são estudados pontos como: Regras formais para obter conclusões válidas; o que pode ser computado; Como raciocinamos com informações incertas. Os filósofos mapearam a maioria das idéias importantes sobre Inteligência Artificial, porém o salto para uma ciência formal exigiu certo nível de formalização matemática em três áreas fundamentais: lógica, computação e probabilidade (RUSSELL; NORVING, 2004).
- **Economia** de 1776 até a atualidade, onde se estuda pontos como: como tomar decisões para maximizar o lucro; como decisões quando os outros não podem nos

acompanhar; como tomar decisões quando o lucro pode estar distante no futuro. A maioria das pessoas imagina que a economia trata de dinheiro, mas na realidade a economia estuda como as pessoas fazem escolhas que levam a resultados preferenciais. O desenvolvimento de teorias na economia contribuiu muito para a noção de agentes racionais, ainda que por muitos anos a pesquisa na Inteligência Artificial tenha se desenvolvido por caminhos completamente distintos. Uma razão para isso era a aparente complexidade das tomadas decisões racionais (RUSSELL; NORVING, 2004).

- **Neurociência** de 1861 até a atualidade, onde é estudado como o cérebro processa informações. A neurociência estuda o sistema nervoso em especial o cérebro. O modo exato como o cérebro habilita o pensamento é um dos grandes mistérios da ciência. Durante milhares de anos observou-se que o cérebro está de alguma forma envolvido no pensamento devido a evidência de que pancadas fortes na cabeça podem levar a incapacitação mental. Sabe-se há muito tempo que o cérebro dos seres humanos tem algumas características diferentes, em aproximadamente 335 a.C., Aristóteles escreveu: “**De todos os animais, o homem é o que tem o maior cérebro em proporção ao seu tamanho.**”. Ainda assim, apenas em meados do século XVIII o cérebro foi amplamente reconhecido como a sede das consciências (RUSSELL; NORVING, 2004).
- **Psicologia** de 1879 até a atualidade, onde é estudado como os seres humanos e animais pensam e agem. A visão do cérebro como um dispositivo de processamento de informações, foi uma característica importante da psicologia cognitiva, o ponto de vista cognitivo foi em grande parte encoberto pelo behaviorismo, que afirmava que qualquer teoria envolvendo processos mentais com base no fato de que a introspecção não poderia fornecer evidência confiável. Somente em 1943 a modelagem cognitiva foi capaz de florescer (RUSSELL; NORVING, 2004).
- **Engenharia de Computadores** de 1940 até a atualidade, onde é estudado como construir um computador eficiente. Para que a Inteligência Artificial tivesse sucesso, era preciso inteligência e um artefato, sendo o computador o artefato preferido. O computador eletrônico digital moderno foi construído quase ao mesmo tempo por países que participavam da segunda guerra mundial. O primeiro computador operacional foi a máquina eletromecânica de Heath Robinson,

construída em 1940 pela equipe de Alan Turing como o único propósito de decifrar mensagens alemãs. No período pós-guerra, Turing queria utilizar os computadores para pesquisas em Inteligência Artificial, mas seus esforços foram bloqueados pelo governo britânico. A área da Ciência da Computação também contribuiu muito com a IA, com o fornecimento de sistemas operacionais, linguagens de programação e as ferramentas necessárias para escrever programas (RUSSELL; NORVING, 2004).

- **Teoria de Controle e Cibernética** de 1948 até a atualidade, onde é estudado como os artefatos podem operar sob seu próprio controle. Em cerca de 250 a.C. foi construída a primeira máquina autocontrolada: um relógio de água com um regulador que mantinha o fluxo de água sempre em um ritmo constante e previsível, a partir daí mudou-se completamente a visão do que um artefato poderia ser capaz de fazer. Antes apenas os seres vivos podiam modificar seu comportamento de acordo com o ambiente (RUSSELL; NORVING, 2004).

Após anos de pesquisas e aprimoramento realizados por estas áreas anteriormente citadas, somente em 1943 iniciou-se a gestação da IA. O primeiro trabalho agora já reconhecido como IA foi realizado por Warren McCulloch e Walter Pitts. Eles basearam-se em algumas fontes: filosofia básica, função dos neurônios no cérebro, análise formal da lógica proposicional e a teoria da computação de Turing. Estes dois pesquisadores propuseram um modelo de neurônios artificiais, com isso eles mostraram que qualquer função computável poderia ser calculada por certa rede de neurônios conectados, que podiam ser implementados por estruturas simples. Também sugeriram que redes definidas adequadamente seriam capazes de aprender (RUSSELL; NORVING, 2004).

Surgiram vários exemplos de trabalhos que hoje podem ser caracterizados como IA, mas foi Alan Turing que, em 1950, articulou uma visão completa da IA com o conhecido teste de Turing.

Somente em 1956 nasce a IA de fato, com a iniciativa de alguns pesquisadores, entre eles John McCarthy, Minsky, Claude Shannon e Nathaniel Rochester, que se reuniram para pesquisar sobre teoria dos autômatos, redes neurais e no estudo da inteligência. Organizaram um seminário com duração de dois meses no Dartmouth College (que se tornou o lugar oficial do nascimento desse campo) no verão de 1956. Dentro desse seminário dois pesquisadores da Carnegie Mellon University (CMU), Allen Newell e Herbert Simon, se destacaram e chamaram muita atenção, pois enquanto muitos tivessem idéias e, em alguns casos programas

para aplicações específicas como jogos de damas, Newell e Simon já tinham um programa de raciocínio onde Simon afirmava-se que programa de computador seria capaz de pensar não - numericamente e assim resolvendo o antigo dilema mente-corpo (RUSSELL; NORVING, 2004). Logo após o seminário o programa foi capaz de demonstrar e provar vários teoremas. Nos vinte anos seguinte, o campo foi dominado por estes pesquisadores, seus alunos e colegas do MIT, CMU, Stanford e IBM.

Apesar do seminário não trazer nenhuma inovação, ele serviu para que as pessoas de maior interesse na área se conhecessem, unindo assim os esforços para tornar a IA uma nova área, já que isto ficou bastante evidente no próprio seminário, a IA já não se encaixava apenas como uma parte de outra área existente, uma vez que a IA é o único campo a tentar construir máquinas que funcionarão de forma autônoma em ambientes complexos e mutáveis.

Os primeiros anos da IA (1952 à 1969) foram de grande entusiasmo e repletos de sucessos, mas com certa limitação, pois os primitivos computadores e as ferramentas de programação da época não permitiam um maior desempenho. A classe intelectual duvidava que uma máquina pudesse reproduzir alguns estados que até então eram realizados apenas por seres humano como: ser amável, ser amigável, ter senso de humor, apaixonar-se, entre outros, onde estes estados eram chamados de "X", mas os pesquisadores da IA respondiam naturalmente demonstrando um "X" após outro.

Apesar das evidências e do esforço dos pesquisadores na área de IA, em meados de 1973, dificuldades começaram a surgir, até então a IA tinha sido utilizada para a resolução de problemas simples e acabaram falhando desastrosamente quando foram realizados experimentos onde o nível de complexidade era mais elevado. A incapacidade de conviver com a "Explosão Combinatória" foi uma das principais críticas a IA, inclusive várias universidades cancelaram o apoio às pesquisas nesta área. Os pesquisadores notaram que um dos grandes problemas nas suas pesquisas era o pouco ou nenhum conhecimento sobre o assunto onde se queria chegar a uma determinada solução, com isso surgiu uma nova metodologia chamada de Sistemas Especialistas, cuja idéia principal era desenvolver sistemas focados em um determinado problema, alimentado por uma fonte de conhecimento especialista existente.

Em 1980, a IA se torna uma indústria. Seu primeiro sistema especialista bem sucedido o "R1", este programa ajudou a configurar pedidos de novos sistemas de computadores, e em 1986 ele estava fazendo a empresa *Digital Equipment Corporatin* (DEC) economizar cerca de 40 milhões de dólares por ano. Em 1988, o grupo de IA da DEC tinha 40 sistemas

especialistas entregues e outros sendo produzidos. De modo geral, a indústria da IA se expandiu de alguns milhões de dólares para bilhões de dólares em 1988. No ano de 1987, a IA se torna uma ciência de fato, embasada pela revolução dos trabalhos de Inteligência Artificial nos últimos anos, tanto no conteúdo quanto na metodologia, de certa forma a IA surgiu como uma resposta as limitações existentes em outras áreas.

Assim, embora a IA seja estudada academicamente desde os anos 50, só recentemente tem gerado um interesse crescente por causa do surgimento de aplicações comerciais e práticas, em vários campos como: Robótica, Reconhecimento de Padrões, Reconhecimento de linguagem e resolução de problemas, Planejamento logístico, Diagnóstico, Controle autônomo, Jogos, entre outros. Um fator decisivo para o sucesso desta transição da academia para a indústria são os enormes avanços tecnológicos dos equipamentos computacionais ocorridos nas últimas duas décadas, possibilitando assim a utilização da IA.

3 INTELIGÊNCIA ARTIFICIAL

A Inteligência Artificial nada mais é que uma forma de fazer uma máquina reagir, interagir e porque não pensar de forma inteligente, ou seja, parte da ciência da computação voltada ao desenvolvimento de sistemas de computadores que demonstrem características associadas à inteligência humana quanto ao seu comportamento, aprendizado, compreensão de linguagens, racionalizando e resolvendo problemas.

A IA imita o processo básico de aprendizado humano por meio da qual novas informações são absorvidas e se tornam disponíveis para referências futuras. A mente humana pode incorporar novos conhecimentos sem alterar seu funcionamento e sem atrapalhar todos os outros fatos que já estão armazenados no cérebro. A definição de IA é bastante ampla e também diversificada, por isso, vejamos algumas definições de IA:

IA é a parte da ciência da computação voltada para o desenvolvimento de sistemas de computadores inteligentes, sistemas que exibem características, as quais nós associamos com a inteligência no comportamento humano, compreensão da linguagem, aprendizado, raciocínio, resolução de problemas etc. (RUSSELL; NORVING, 2004, p.26).

“É o campo de estudo que tenta explicar e emular o comportamento inteligente em termos de processos computacionais.” (SCHALKOFF, 1990, p.18).

“O ramo da ciência da computação preocupada com a automação de comportamento inteligente.” (LUGER; STUBBLEFIELD, 1993, p.12)

“Inteligência Artificial é o estudo das idéias que permitem aos computadores serem inteligentes” (WINSTON, 1984, p.43).

3.1 ABORDAGENS DA INTELIGÊNCIA ARTIFICIAL

Entre as técnicas de IA disponíveis nos dias de hoje, cabe a cada projetista escolher aquela que seja mais adequada ao seu problema. Alguns fatores como a disponibilidade de dados e a fundamentação teórica são de extrema importância na escolha do método. Baseado nos diversos campos de estudo que ajudaram na fundamentação dos princípios teóricos da inteligência artificial surgiram duas abordagens com maior evidência:

- **Teoria Cognitiva** - Também conhecida como descendente ou simbolista, destaca processos cognitivos, ou melhor, a forma como pensa o ser humano. Tem como objetivo procurar um esclarecimento para comportamentos inteligentes que se baseiam em processos algorítmicos e aspectos psicológicos. Os primeiros a participar dessa vertente foram Newell, Marvin Minsky, John McCarthy e Simon. Os primeiros protótipos de modelos baseados na inteligência apareceram na década de 50 e obtiveram como referência as regras de produção e a lógica dos predicados. Outro acontecimento importante, que se deu pela consequência da formalização da lógica, foi a facilitação do processo de formalização e representação dos conhecimentos a serem utilizados pelos programas de computador. Primordialmente, esses conhecimentos se reduziam a simples esquemas de raciocínios para jogos, aplicações matemáticas e simuladores (RUSSELL; NORVING, 2004).
- **Teoria Conexionista** - Também chamada de biológica ou ascendente, dá ênfase no modelo de funcionamento do cérebro, dos neurônios e das conexões neurais. Os pioneiros dessa corrente foram Widrow, Hebb, McCulloch, Rosenblatt e Pitts. Em 1943, surgiram, a partir da representação e formalização matemática dos neurônios artificiais, os primeiros modelos de redes neurais artificiais. A corrente conexionista sofreu grande impacto quando os cientistas Seymour Papert e Marvin Minsky publicaram (1969) o livro *Perceptrons*, no qual criticaram e sustentaram

que os modelos das redes neurais não tinham sustentação matemática suficiente que lhes fosse possível atribuir alguma confiabilidade. Apesar das pesquisas nesta área não terem parado, foi apenas na década de 80 que, o físico e biólogo do Instituto de Tecnologia da Califórnia, Jonh Hopfield conseguiu recuperar a credibilidade da utilização das redes neurais (RUSSELL; NORVING, 2004).

3.2 CAMPOS DE APLICAÇÃO DA INTELIGÊNCIA ARTIFICIAL

Existem inúmeros campos de estudo dentro da IA com o propósito de dotar a máquina de capacidade de raciocínio, aprendizado e auto-aperfeiçoamento, alguns desses campos são descritos abaixo:

- **Processamento de Linguagem Natural** - É o estudo voltado para a construção de programas capazes de compreender a linguagem natural (interpretação) e gerar textos. A Geração de linguagem Natural é a produção de textos por um programa a partir de um conteúdo semântico representado internamente no próprio programa. Objetiva aperfeiçoar a comunicação entre as pessoas e os computadores (RUSSELL; NORVING, 2004).
- **Reconhecimento de Padrões** - É uma das áreas de pesquisa bem avançadas da IA. A capacidade de reconhecimento de padrões permite ao programa reconhecer a fala em linguagem natural, os caracteres digitados e a escrita (ex.: assinatura). Os scanners, por exemplo, utilizam programas de reconhecimento óptico desenvolvidos pelas pesquisas em IA (RUSSELL; NORVING, 2004).
- **Visão de Computador** - Busca desenvolver formas para o computador trabalhar com a visão bidimensional e tridimensional (RUSSELL; NORVING, 2004).
- **Programação de Jogos** - É o estudo voltado para a construção de programas de jogos envolvendo raciocínio. Os jogos computadorizados são um grande sucesso, ainda mais quando exibem um tipo de inteligência capaz de desafiar as habilidades do jogador. O jogo de xadrez, por exemplo, foi utilizado para as primeiras experiências em programação do raciocínio artificial, onde o computador se tornou capaz de analisar milhões de jogadas por segundo para tentar derrotar o adversário.

Além de analisar as jogadas, os programas utilizam um método heurístico que consiste na utilização de uma árvore de busca, a mesma possui ramificações a partir de certos nós, que representam pontos de decisão no caminho a tomar, com certo número de etapas, para chegar a um objetivo. Deste modo, ele pode analisar vários nós e, de acordo com a situação atual do jogo, escolher o melhor caminho (o mais curto ou menos arriscado) (RUSSELL; NORVING, 2004).

- **Robótica** - É o campo de estudo voltado para desenvolver meios de construir máquinas que possam interagir com o meio (ver, ouvir e reagir aos estímulos sensoriais). A expressão robô vem do tcheco *robota*, significa trabalhador, foi criada por Karel Capek, em 1917. O primeiro robô industrial do mundo, batizado de UNIMATE, surgiu em 1962 (RUSSELL; NORVING, 2004).
- **Aprendizado** - Existem programas de IA que conseguem aprender certos fatos por meio da experiência, desde que esse conhecimento possa ser representado de acordo com o formalismo adotado pelo programa (RUSSELL; NORVING, 2004).

3.3 MODELOS DE INTELIGÊNCIA ARTIFICIAL

Alguns modelos destacam-se bastante na IA, pelo simples fato de se adequarem melhor a um determinado problema, conseqüentemente, são bastante utilizados, tais como:

- **Algoritmos Genéticos** - É um modelo para o aprendizado da máquina, inspirado no livro *Origem das Espécies*, através da Seleção Natural, escrito pelo naturalista inglês Charles Darwin (1809-1882), criador da teoria evolucionista, segundo a qual somente os mais aptos sobrevivem. Algoritmo genético é um método utilizado pelos Algoritmos Evolutivos que inclui o estudo dos algoritmos genéticos, estratégia de evolução, programação evolutiva e sistemas classificatórios. Os Algoritmos Genéticos, criados por Jonh Holland (1975), objetivam emular operadores genéticos (específicos, como *crossing-over*, mutação e reprodução) da mesma forma como é observado na natureza. Isso é feito criando-se dentro da máquina uma população de indivíduos representados por cromossomas. Os

indivíduos passam por um processo simulado de evolução, seleção e reprodução, gerando uma nova população (RUSSELL; NORVING, 2004).

- **Programação Evolutiva** - Campo da IA concebido por Lawrence J. Fogel (1960) assemelha-se aos algoritmos genéticos, sendo que é dada maior ênfase na relação comportamental entre os parentes e seus descendentes. As soluções para os problemas são obtidas por meio tentativas e transmitidas para a nova população (simulada em programas) (RUSSELL; NORVING, 2004).
- **Lógica FUZZY** - Também denominada de Conjuntos Difusos ou Lógica Nebulosa. Foi estruturada por Lofti Zadeh da University of Califórnia, no ano de 1965. É uma metodologia que serve para representar, manipular e modelar informações incertas (RUSSELL; NORVING, 2004).
- **Sistemas Baseados em Conhecimento** - São sistemas que implementam comportamentos inteligentes de especialistas humanos (RUSSELL; NORVING, 2004).
- **Programação Genética** - É um campo de estudo da IA voltado para a construção de programas que visam imitar o processo natural da genética. Trabalha com métodos de busca aleatória (RUSSELL; NORVING, 2004).
- **Raciocínio Baseado em Casos** - É o campo de estudo da IA que utiliza uma grande biblioteca de casos para consulta e resolução de problemas. Os problemas atuais são resolvidos através da recuperação e consulta de casos já solucionados e da conseqüente adaptação das soluções encontradas. Por exemplo, o Sistema CASEY que faz o diagnóstico em pacientes cardíacos baseado na consulta de arquivos de pacientes com o mesmo diagnóstico (RUSSELL; NORVING, 2004).
- **Redes Neurais Artificiais (RNA)** - Possui várias denominações, dentre elas redes neurais, modelo connexionista, neurocomputação, modelo de processamento paralelo distribuído, sistemas neuromórficos e computadores biológicos (RUSSELL; NORVING, 2004).

Existem outras áreas de estudo que são englobadas no desenvolvimento da IA. Além disso, por ser uma ciência relativamente nova e com um potencial muito grande, e ainda inexplorado, possivelmente outras áreas de pesquisa e de aplicações surgirão nas próximas décadas.

As aplicações desenvolvidas através da IA têm como característica principal a interação com o usuário ou seu ambiente real, por meio da emulação de reações humanas que englobam a análise, planejamento, aprendizado, tomada de decisão, compreensão, etc.

Estas aplicações são utilizadas com eficiência considerável, substituindo ou provendo suporte ao homem em determinadas tarefas de aprendizagem, como: reconhecimento de padrões, tradução, reconhecimento de padrões de imagens, diagnóstico médico, previsões econômicas, reconhecimento de voz, controle de qualidade, automação de processos industriais, etc.

3.4 - FUNDAMENTAÇÃO BIOLÓGICA

O cérebro é especialista em desempenhar tarefas altamente complexas, entretanto o cérebro humano também realiza generalizações incorretas, e pode algumas vezes ser impreciso lento e, na maioria das vezes, é incapaz de explicar suas próprias ações. Apesar do grande interesse e pesquisas desenvolvidas acerca do cérebro humano, pouco se sabe sobre este intrigante órgão, e essa imprecisão é um dos fatores mais críticos a ser enfrentado. Estar diante de uma máquina bastante complexa, cuja maior dificuldade é desvendar o seu funcionamento.

Sabe-se que em um cérebro humano contém aproximadamente 10 (dez) bilhões, de unidades básicas, que são chamadas de neurônios, onde cada neurônio está conectado a cerca de 10 (dez) milhões de outros neurônios. Os neurônios biológicos são conectados aos outros neurônios através de ligações sinápticas, acredita-se que a aptidão das sinapses serem moduladas é a base fundamental para todos os processos cognitivos, como memória, percepção e raciocínio.

Três partes compõem os neurônios, são elas: axônio, dendritos e corpo celular, como podemos observar na Figura 01. Pelos dendritos é feita a condução das informações recebidas de outros neurônios, as informações são somadas no corpo celular do neurônio, enviando ou não uma nova informação a outros neurônios que por sua vez deixará a célula através do axônio como informação para outros neurônios.

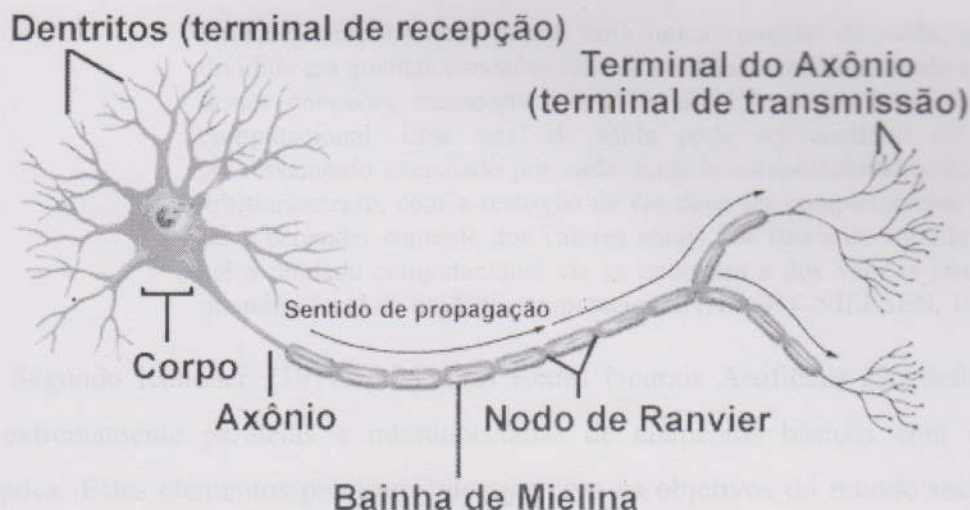


Figura 01: Neurônio Biológico
Fonte: Neves (2004).

Estruturas multicelulares indispensavelmente possuem um tipo de sistema nervoso. Até os animais de organismos mais básicos, tem a capacidade de adaptar o seu comportamento e registrar informações em seus sistemas nervosos. A todo instante o organismo recebe comandos do sistema nervoso, por meio de entradas sensoriais, a respeito das informações sobre a situação do ambiente no qual ele se encontra. No cérebro os neurônios processam as informações de entrada que são confrontadas com as experiências anteriores e transformada, posteriormente, em ações adequadas ou absorvidas sob o formato de conhecimento.

4 REDES NEURAIS ARTIFICIAIS

Podemos chegar a várias definições do que venha a ser uma Rede Neural Artificial (RNA). O neurônio é a unidade computacional fundamental da rede em questão, onde em uma RNA existe uma arquitetura, que é a estrutura topológica de como os neurônios desta RNA são conectados, e também a aprendizagem, procedimento que adapta a RNA de modo a calcular uma função esperada ou desempenhar uma tarefa. A seguinte descrição formal de uma RNA foi proposta:

Uma Rede Neural Artificial é: uma estrutura que processa informação de forma paralela e distribuída e que consiste de unidades computacionais (as quais podem possuir uma memória local e podem executar operações locais) interconectadas por canais unidirecionais chamados de conexões. Cada

unidade computacional possui uma única conexão de saída, que pode ser dividida em quantas conexões laterais se fizer necessário, sendo que cada uma destas conexões transporta o mesmo sinal, o sinal de saída da unidade computacional. Esse sinal de saída pode ser contínuo ou discreto. O processamento executado por cada unidade computacional pode ser definido arbitrariamente, com a restrição de ele deve ser completamente local, isto é, deve depender somente dos valores atuais dos sinais de entrada que chegam até a unidade computacional via as conexões e dos valores armazenados na memória local da unidade computacional. (HECHT-NIELSEN, 1990, p.181).

Segundo Kohonen (1972, p.358), as Redes Neurais Artificiais são definidas como redes extremamente paralelas e interconectadas de elementos básicos com organização hierárquica. Estes elementos precisam interagir com os objetivos do mundo real da mesma forma que o sistema nervoso biológico.

Para Lippmann (1997, p.68), as RNA's são sistemas físicos capazes de adquirir, registrar e utilizar os conhecimentos experimentais que poderão conseguir um bom desempenho devido à sua forte interconexão em meio aos nós da rede. Essas são conhecidas também por: modelos de processamento paralelo distribuído, modelos conexionistas e sistemas neuromorfológicos.

Nos dias de hoje, cada vez mais profissionais de outras áreas se interessam por RNA's, pela sua grande capacidade de interação de acordo com o ambiente. Os psicólogos, por exemplo, interessam-se pela semelhança estrutural das RNA's em relação à mente humana, já os pesquisadores da área de IA buscam nesses modelos soluções para um dos grandes problemas desta área: o aprendizado de máquina.

4.1 O NEURÔNIO DE MCCULLOCH E PITTS

O modelo mais clássico de neurônio em RNA's foi proposto por McCulloch e Pitts em 1943. Eles afirmaram que o neurônio está executando uma função lógica. Este trabalho, intitulado "Um cálculo lógico das idéias intrínsecas da atividade neural" é bastante ilustre e exerceu ampla influência na sociedade científica da época. Foi o primeiro experimento a entender a atividade neural fundamentado em unidades simples de computação. A partir daí surgiram cinco premissas:

- A atividade do neurônio é binária, podendo apresentar apenas dois estados lógicos: 1 (ativo) ou 0 (inativo);
- Uma certa quantidade fixa de sinapses deve ser excitada em um período latente de adição de forma a excitar o neurônio e este número é independente da atividade prévia e da posição do neurônio;
- O único atraso significativo no cérebro é o atraso do processamento sináptico;
- A atividade de uma sinapse inibitória bloqueia completamente a atividade do neurônio num dado instante de tempo;
- A estrutura da rede neural não varia com o tempo.

Segundo Russell e Norving (2004, p.102) McCulloch e Pitts avaliaram a resposta neural como sendo equivalente a uma conjuntura (se A então B) apropriada para estimular o neurônio. Sendo assim, eles analisaram o comportamento de várias RNA's utilizando a notação simbólica da lógica das proposições. A propriedade binária dos neurônios era suficiente para garantir que a atividade de qualquer neurônio teria a capacidade de ser expressa sob a visão da lógica das proposições.

Mesmo com o conhecimento moderno sobre a estrutura física e química que envolve o envio de sinais para o cérebro seja plausível para assegurar que nenhum tipo de lógica das proposições é desempenhada no cérebro, o modelo do neurônio de McCulloch e Pitts pode ser aceito como um caso particular do neurônio genérico utilizado em RNA's. Considerando a representação simbólica do neurônio de McCulloch e Pitts, Figura 02:

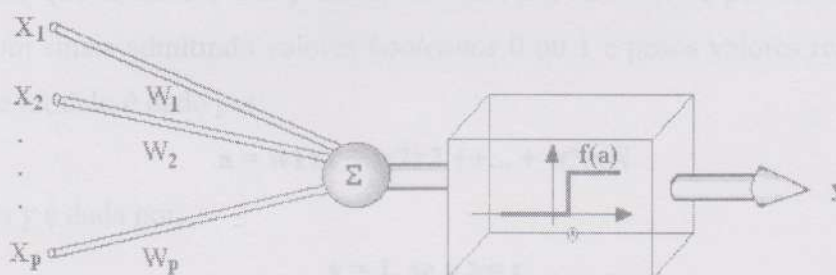


Figura 02: Representação do neurônio de McCulloch e Pitts.
Fonte: Mendes (1998).

Para toda iteração t , o neurônio responde através das suas entradas sinápticas, que influenciam no estado do neurônio pré-sináptico. Se nenhuma sinapse inibitória está ativa, o neurônio agrega (soma) suas entradas, determinando a entrada u do neurônio e analisa se u é maior do que o limiar. Caso isto seja verdade, o neurônio envia um sinal de saída com o valor 1. Caso isto seja negativo, ele envia um sinal de saída com o valor 0.

4.2 CARACTERÍSTICAS GERAIS DAS REDES NEURAIS

Uma RNA é necessariamente constituída por várias unidades de processamento que possuem um funcionamento bastante simples. Entretanto, estas unidades são comumente ligadas por canais de comunicação que estão conectados a um determinado peso. Estas unidades fazem operações somente sobre seus dados locais, que são dados recebidos pelas suas ligações. Segundo Gurney (1997), a conduta inteligente de uma RNA vem das conexões e intercâmbio entre as unidades de processamento da rede.

A operação de uma unidade de processamento, que pode ser observada na Figura 02, é enfatizada por McCullock e Pitts em 1943 (ACKLEY, 1985), e resume-se da seguinte maneira:

- Sinais são apresentados à entrada;
- Cada sinal é multiplicado por um número, ou peso, que indica a sua influência na saída da unidade;
- É feita a soma ponderada dos sinais que produz um nível de atividade;
- Se este nível de atividade exceder certo limite, a unidade produz uma determinada resposta de saída.

Supondo que tenhamos tais p sinais de entrada $x_1, x_2, \dots, x_N$ e pesos $w_1, w_2, \dots, w_N$ e limitador t , com sinais admitindo valores *booleanos* 0 ou 1 e pesos valores reais. O nível de atividade neste modelo é dado por:

$$a = w_1x_1 + w_2x_2 + \dots + w_Nx_N$$

A saída y é dada por:

$$y = 1, \text{ se } a \geq t \\ \text{ou} \\ y = 0, \text{ se } a < t$$

A grande maioria dos modelos de redes neurais possui determinada regra de treinamento, onde os pesos de suas conexões são corrigidos ou ajustados de acordo com padrões apresentados. De forma simplória, o aprendizado se dá através de exemplos.

Estruturas de RNA's são caracteristicamente organizadas em camadas, com unidades que podem estar conectadas às unidades da camada posterior, como podemos observar na Figura 03.

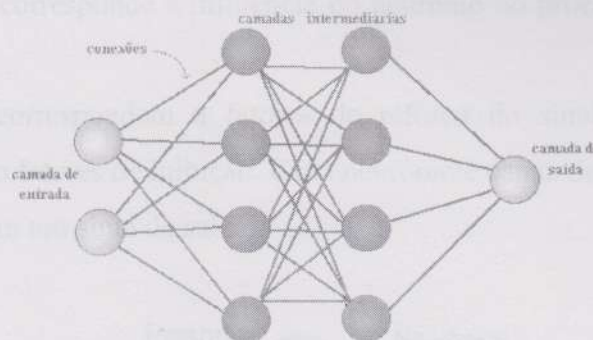


Figura 03: Camadas de uma Rede Neural Artificial.

Fonte: Neves (2004).

Habitualmente, as camadas são classificadas em três grupos:

- **Camada de Entrada:** onde é feita a apresentação dos padrões a RNA.
- **Camadas Intermediárias ou Escondidas:** a maior parte do processamento é feita nesta camada, através das conexões ponderadas, podem ser consideradas como extratoras de características.
- **Camada de Saída:** onde é apresentado o resultado final.

Um neurônio soma todos os pesos das entradas passa o resultado para uma função de ativação não-linear. Há três tipos de não-linearidade: Limitadores elevados; elementos limiares lógicos e não linearidades sigmóides (LIPPMANM, 1997).

As RNA's podem ser distinguidas com base na direção na qual o sinal flui. Basicamente, há dois tipos de RNA's: *feedforward* e *feedback*. Nas RNA's *feedforward* os sinais propagam-se apenas em uma direção a partir da unidade de entrada, passando pela camada intermediária até a saída. Um caso extremo de uma RNA *feedforward*, é a rede inteiramente ligada, onde cada neurônio está ligado a todo o aglomerado de neurônios, onde toda estrutura de interconexão está ligada. Nas RNA's do tipo *feedback*, os sinais de entrada podem se propagar da saída de qualquer neurônio para a entrada em um outro neurônio.

Uma RNA é especificada, especialmente, pela sua topologia, pelas características dos nós e pelos modos de treinamento. Segundo Kohonen (1972, p.356), um modelo de RNA's consiste em uma rede de neurônios, relativamente independentes, dotados de capacidade de processamento.

Os modelos na maioria das vezes apresentam um conjunto de neurônios de entrada, por onde são fornecidas as informações para a rede, e um conjunto de neurônios de saída, que

representam os sinais de saída da RNA, e um conjunto de neurônios intermediários como observa-se na Figura 04.

Para Kohonen (1972, p.353), os neurônios são ligados por conexões cada um com um peso (w) associado, que corresponde à influência do neurônio no processamento do sinal de saída.

Pesos positivos correspondem a fatores de reforço do sinal de entrada e pesos negativos correspondem a fatores de inibição. Cada neurônio é capaz de processar um sinal de entrada e transformá-lo em um sinal de saída.

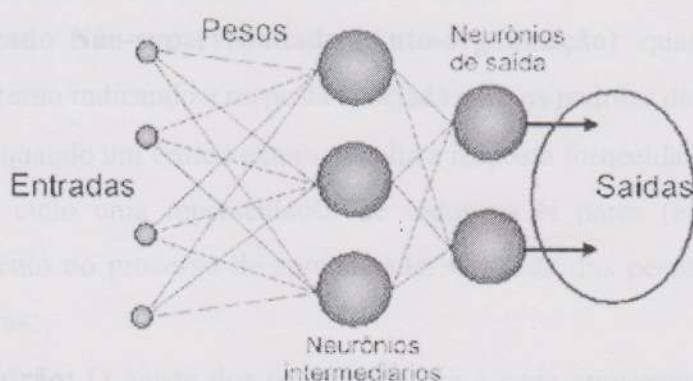


Figura 04: Estrutura de modelo de RNA's.

Fonte: Mendes (1998).

Uma das propriedades mais importantes das RNA's é a capacidade de "aprender sobre seu ambiente" e, com isso, melhorar seu desempenho.

Segundo Ackley (1985, p.149), a capacidade de aprender ocorre através de um processo iterativo de adaptação aplicada a seus pesos, o treinamento. O aprendizado ocorre quando uma rede neural alcança uma solução generalizada para uma classe de problemas, atividade que observa-se na Figura 05.

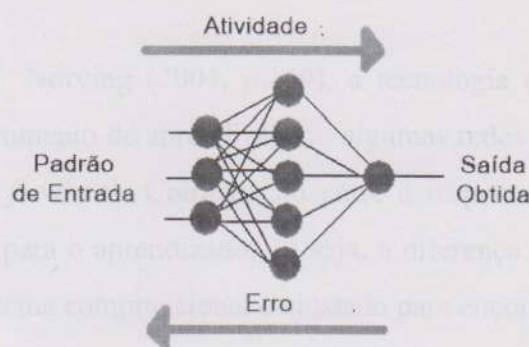


Figura 05: Aprendizagem de uma Rede Neural Artificial.

Fonte: Neves (2004).

Chama-se de algoritmo de aprendizado o conjunto de regras bem definidas para a solução de um problema de aprendizado. Existem muitos tipos de algoritmos de aprendizado específicos para modelos de RNA's em específico que diferem entre si principalmente pelo modo como os pesos são alterados.

Outro fator de grande importância é a maneira pela qual uma RNA se relaciona como ambiente. Os seguintes paradigmas de aprendizado existem neste contexto:

- **Aprendizado Supervisionado:** quando é utilizado um agente externo que indica à rede a resposta desejada para o padrão de entrada;
- **Aprendizado Não-supervisionado (Auto-organização):** quando não existe um agente externo indicando a resposta desejada para os padrões de entrada;
- **Reforço:** quando um crítico externo avalia a resposta fornecida pela rede.

Denomina-se ciclo uma representação de todos os N pares (entrada e saída) do conjunto de treinamento no processo de aprendizado. O ajuste dos pesos no ciclo pode ser feitas de duas maneiras:

- **Modo padrão:** O ajuste dos pesos acontece a cada apresentação à RNA de um exemplo do conjunto de treinamento. Cada ajuste de pesos baseia-se somente no erro do exemplo apresentado naquela iteração. Assim, em cada ciclo ocorrem N correções.
- **Modo Batch:** Apenas um ajuste é feito por ciclo. Todos os exemplos do conjunto de treinamento são apresentados à RNA, seu erro médio é calculado e a partir desse erro fazem-se os ajustes dos pesos.

Para Russell e Norving (2004, p.137), uma RNA não executa uma série de instruções, ela responde em paralelo as entradas apresentadas. O resultado não é armazenado em um local específico da memória, mas consiste em todos os estados da rede, após encontrar uma condição de equilíbrio.

Segundo Russell e Norving (2004, p.149), a tecnologia de redes neurais também utiliza *feedback* como instrumento de aprendizagem, algumas redes neurais aprendem usando a resposta desejada como *feedback*. Comparação entre a resposta computada e a resposta desejada vem a ser a base para o aprendizado, ou seja, a diferença entre as duas é repassada para trás no sistema e o sistema computacional é ajustado para encontrar a resposta correta.

Uma vez definido um neurônio, é possível pesquisar as propriedades das RNA's. Observando uma RNA do ponto de vista da entrada e saída de informações, ela pode conter

três tipos de neurônios: neurônio de entrada - u_k -, neurônio de saída - y_k -, e finalmente os neurônios internos. Há um interesse biológico bastante particular neste último tipo de neurônio, pelo simples fato de que, admitindo ciclos na sua estrutura de interconexão, é possível que ele desenvolva atividades sem depender de estímulos externos. Interpretam-se essas ações autônomas como “pensamentos”. Os neurônios da camada interna, também são bastante importantes do ponto de vista matemático, pois é provável que a classe de problemas conhecidos como não linearmente separáveis, do qual podemos dizer que o “ou - exclusivo” é uma instância, só consente uma solução por meio de RNA's que tenham pelo menos uma camada interna.

Enxergando pela ótica funcional, uma RNA pode ser homogênea se o comportamento de todos os neurônios da RNA em questão forem iguais, ou, caso contrário, heterogêneas. As RNA's são na maioria das vezes homogêneas.

O primeiro modelo de rede RNA, foi chamado *Perceptron*, foi sugerido por Rosenblatt, em 1957. Ele consiste em uma RNA de duas camadas, uma de entrada: utilizada somente para remanejamento da informação de entrada, e outra de saída: onde é realizado o processamento de fato, esta camada é constituída por neurônios binários. Este modelo serviu de alavanca para outros modelos subseqüentes, pois após seu desenvolvimento diversos modelos foram idealizados. Alguns destes modelos, apresentados na Tabela 01, são mais conhecidos pela sua eficácia, funcionalidade, vantagens e desvantagens.

Modelos de Redes Neurais

Redes	Aplicações	Vantagens	Desvantagens
Perceptrons	Reconhecimento de caracteres	Rede neuronal mais antiga	Não reconhece padrões complexos, sensível a mudanças
Backpropagation	Larga aplicação	Rede mais utilizada, simples e eficiente	Treinamento supervisionado exige muitos exemplos
Counterpropagation	Reconhecimento de padrões, análise estatística	Rapidez do treinamento	Topologia complexa
Hopfield	Recuperação de dados e fragmentos de imagens	Implementação em larga escala	Sem aprendizado, pesos preestabelecidos
Bidirectional Associative Memories (BAM)	Reconhecimento de padrões	Estável	Pouco eficiente
Kohonen	Reconhecimento de padrões não especificados	Auto-organização	Pouco eficiente

Quadro 1

Fonte: Rabuske (1995).

Independente da estrutura de uma RNA, ao longo do tempo seu comportamento pode ser matematicamente formalizado por meio de instrumentos formais da álgebra linear. A cada momento, o estado de uma RNA, com n neurônios, é representado, por um vetor n -dimensional onde os componentes são os valores das ações neuronais x_i . Contudo, é feita a associação do vetor aos níveis de ativação σ_i dos neurônios. As ligações entre neurônios, com seus relativos pesos, poderão ser representadas no formato de uma matriz $N \times N$, na ausência de conexão os pesos terão valor zero. Com este perfil, pode-se obter os níveis de ativação do vetor, em um dado momento do tempo, a partir do vetor de atividades associado ao instante anterior através do seguinte produto matricial:

$$\sigma(t+1) = [W_{ij}(t)] x(t).$$

Uma RNA é um sistema mutável onde o novo estado é a função do estado que acabou de ser alterado. Atualização é o cálculo do novo estado da RNA, sendo esta denominada de duas formas:

- **Síncrona:** Quando os novos níveis de acionamento são determinados ao mesmo tempo, em função dos valores imediatos das atividades.
- **Assíncrona:** Em que é atualizado o nível de acionamento de uma única unidade por vez.

4.3 UNIDADES GENÉRICAS EM RNA'S

Um neurônio biológico funciona da seguinte forma: os sinais de entrada passam através de canais localizados nas sinapses (que são as entradas), permitindo assim a entrada e saída de íons. Este sinal é processado e um potencial de membrana aparece como resultado da união dos sinais de entrada, que ou determinam se o neurônio produzirá um sinal de saída (pulso) ou não. O pulso ocasiona a liberação de neurotransmissores na sinapse, sempre que o potencial de membrana for maior que um determinado limite. Todos estes processos biológicos ocorridos nas sinapses são representados por um **peso** associado. Na maioria das RNA's existe um elemento computacional básico que é um integrador, elemento processador de informações indispensável para o funcionamento das RNA's. As partes mais importantes do neurônio artificial genérico são:

- As sinapses, reconhecidas pelos seus pesos associados;
- A junção somadora;
- A função de ativação.

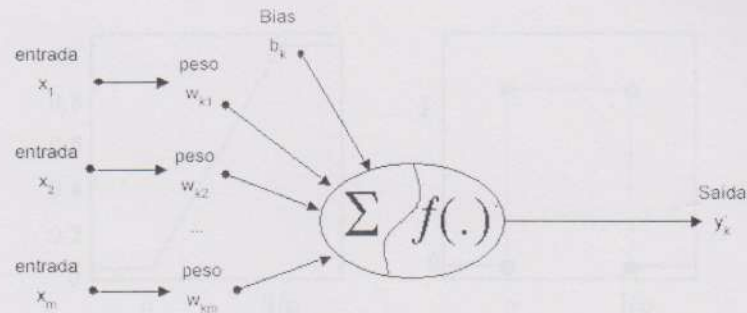


Figura 06: Neurônio artificial genérico.
Fonte: Nascimento; Yoneyama (2004).

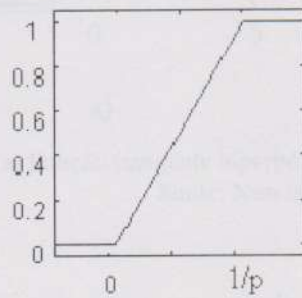
Podemos observar na Figura 06 que o primeiro k do peso sináptico w_{kj} é correspondente ao neurônio pós-sináptico, e o próximo corresponde à sinapse conectada a ele. Todos os sinais de entrada são somados na junção somadora, onde são ponderados pelos pesos das conexões. Os vetores de entrada de pesos assumem como sendo vetores coluna, este processo corresponde ao resultado do vetor de entradas $\mathbf{x}$ pelo vetor de pesos $\mathbf{w}_k$, somado ao limiar b_k . Simplesmente, trata-se de uma junção linear das entradas pelos pesos associados, mais o limiar b_k . A ativação é feita por uma função de ativação utilizada com dois objetivos: racionalizar a saída do neurônio e gerir não-linearidade no modelo. O limiar b_k tem a função de aumentar ou diminuir a força do valor da entrada para a ativação do neurônio k . A saída do neurônio k pode ser formalizada matematicamente (como pode-se observar nas Figuras 07, 08 e 09), por:

$$y_k = f(u_k) = f\left(\sum_{j=1}^m w_{kj} x_j + b_k\right) \quad \text{ou} \quad y_k = f(u_k) = f\left(\sum_{j=0}^m w_{kj} x_j\right)$$

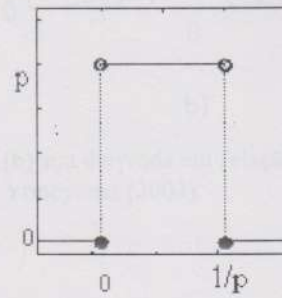
Onde x_0 é um sinal de entrada de valor 1 e peso associado $w_{k0} = b_k$. Exemplos de função de ativação:

$$f(u_k) = \begin{cases} 1 & \text{se } pu_k \geq 1 \\ pu_k & \text{se } 0 < pu_k < 1 \\ 0 & \text{se } pu_k \leq 0 \end{cases}$$

com o p constante e positivo.



a)

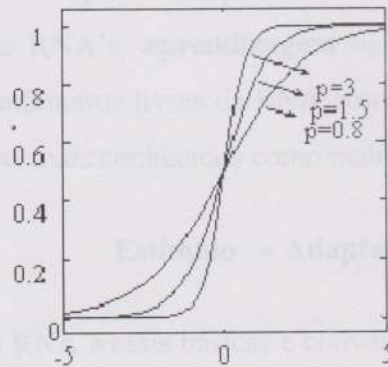


b)

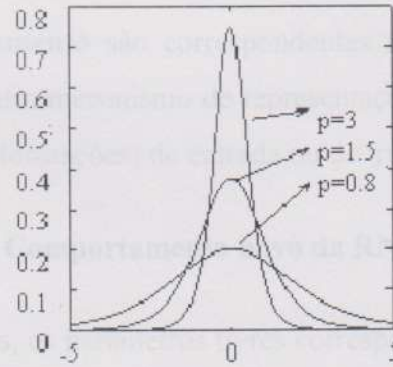
Figura 07: (a) Função (semi-linear) e (b) derivada em relação à entrada interna.

Fonte: Nascimento; Yoneyama (2004).

$$y = f(u_k) = \frac{e^{pu_k}}{e^{pu_k} + 1} = \frac{1}{1 + e^{-pu_k}} \quad \frac{\partial y}{\partial u_k} = pu_k(1 - u_k) > 0$$



a)



b)

Figura 08: (a) Função (logística) e (b) derivada em relação à entrada interna.

Fonte: Nascimento; Yoneyama (2004).

$$y = f(u_k) = \tanh(pu_k) = \frac{e^{pu_k} - e^{-pu_k}}{e^{pu_k} + e^{-pu_k}} \quad \frac{\partial y}{\partial u_k} = p(1 - u_k^2) > 0$$

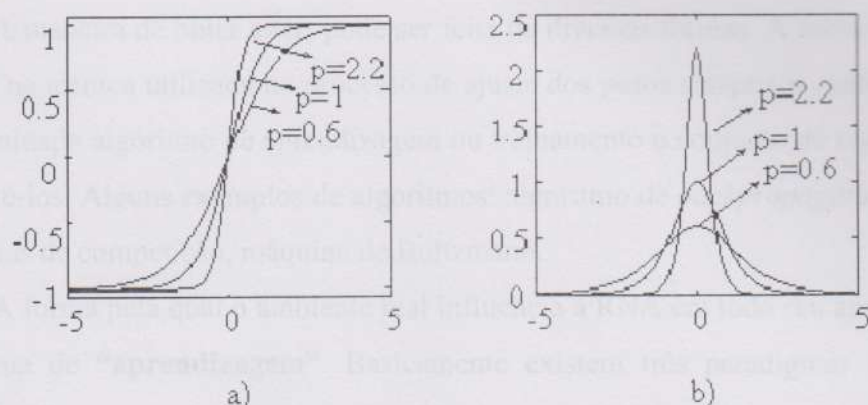


Figura 09: (a) Função (tangente hiperbólica) e (b) sua derivada em relação à entrada interna.
Fonte: Nascimento; Yoneyama (2004).

4.4 ALGUNS PARADIGMAS DE APRENDIZAGEM

A capacidade de aprender é uma das mais importantes qualidades contidas na estrutura de uma RNA. A habilidade de adaptar-se ao ambiente, de acordo com regras existentes, mudando seu desempenho ao longo do tempo. Considerando assim, "aprendizado" o processo que molda e adapta o comportamento, conduzindo a uma melhora de desempenho. Neste contexto de RNA's, **aprendizagem** ou **treinamento** são correspondentes ao processo de ajuste de parâmetros livres da RNA através de um mecanismo de representação de estímulos no ambiente, mais conhecidos como padrões (informações) de entrada ou de treinamento:

Estímulo → Adaptação → Comportamento novo da RNA.

Nas RNA's mais básicas e convencionais, os parâmetros livres correspondem somente aos pesos sinápticos. A estrutura da rede como um todo, incluindo: tipos de neurônios e suas funções de ativação são pré-definidas.

O grande objetivo do aprendizado em RNA's é conseguir de um modelo implícito do sistema em estudo por ajuste dos parâmetros e pesos da rede. A partir de uma RNA, seja $w(t)$ um peso sináptico de um certo neurônio, no momento de tempo t . O ajuste $\Delta w(t)$ é aplicado ao peso sináptico $w(t)$ no momento t , que gerará o valor corrigido $w(t+1)$, na forma: $w(t+1) = w(t) + \Delta w(t)$.

A maneira de obter $aw(t)$ pode ser feita de diversas formas. A forma de aprendizado é baseada na técnica utilizada no processo de ajuste dos pesos sinápticos (parâmetros da RNA). É denominado algoritmo de aprendizagem ou treinamento o conjunto de regras bem definidas para obtê-los. Alguns exemplos de algoritmos: algoritmo de *backpropagation*, regra de Hebb, estratégias de competição, máquina de Boltzmann.

A forma pela qual o ambiente real influencia a RNA em todo seu aprendizado define o paradigma de “**aprendizagem**”. Basicamente existem três paradigmas de aprendizagem: **aprendizado por reforço, aprendizado supervisionado e aprendizado não-supervisionado**.

Este trabalho irá abordar a evolução de técnicas para aprendizado não-supervisionado e supervisionado e em RNA's. E pelo simples fato de serem intuitivas, as técnicas de aprendizado supervisionado serão abordadas primeiramente.

4.5 APRENDIZAGEM COM SUPERVISÃO

O Treinamento com supervisão do modelo de rede *perceptron* consiste em ajustar os pesos e os *thresholds* de suas unidades para que a classificação desejada seja obtida. Para a adaptação do *threshold* juntamente com os pesos, pode-se considerá-lo como sendo o peso associado a uma conexão, cuja entrada é sempre igual a -1 e adequar o peso relativo a essa entrada (TAFNER; XEREZ; RODRIGUES FILHO, 1995).

Quando um determinado padrão é inicialmente apresentado à rede, ela produz uma saída. Após mensurar a distância entre a resposta atual e a esperada, são realizados os ajustes apropriados nos pesos das conexões de modo a encurtar essa distância (FAUSETT, 1994). Este processo é conhecido como Regra Delta.

Desse modo, tem-se o seguinte esquema de treinamento, Figura 10:

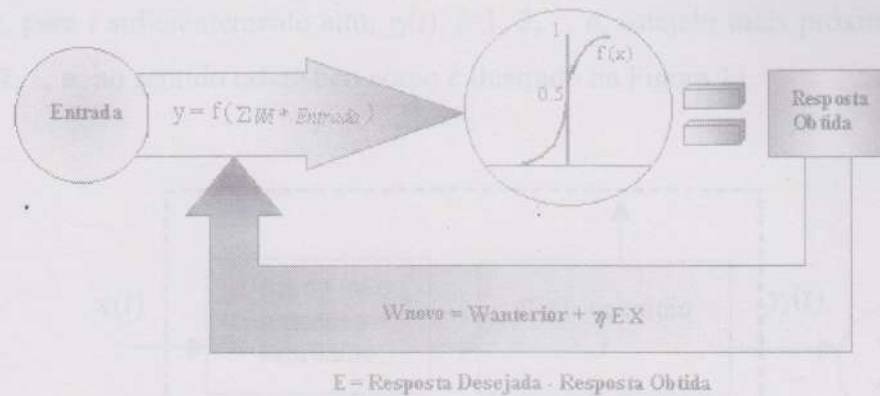


Figura 10: Esquema treinamento do *Perceptron*.
Fonte: Mendes (1998).

Alguns exemplos de problemas de engenharia que podem ser mostrados sob forma de um problema de aprendizagem com supervisão:

- Predição de séries temporais;
- Identificação de sistemas;
- Controle de processos;
- Projeto de filtros em processamento de sinais;
- Classificação e reconhecimento de padrões;

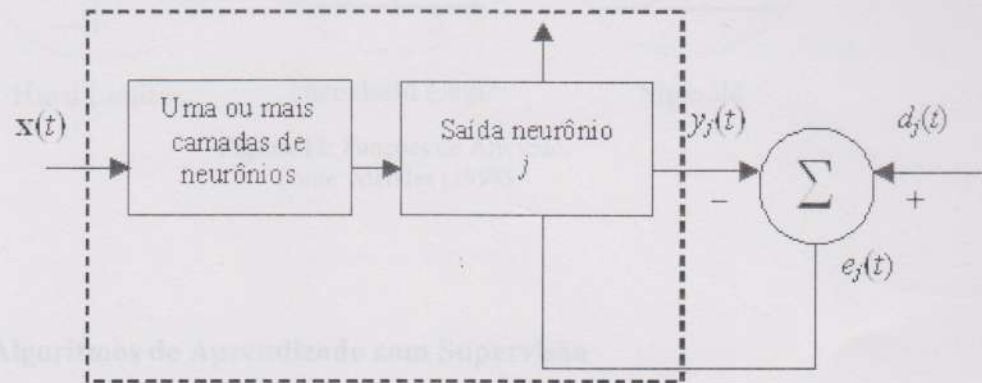
4.5.1 Formalização Processo de Aprendizado com Supervisão

Seja $d_j(t)$ a solução esperada para o neurônio j no momento t e $y_j(t)$ a resposta obtida do neurônio j no momento t , obtida através de um estímulo $x(t)$ presente na entrada da rede neural.

$x(t)$ e $d_j(t)$ constituem o exemplo de estímulo-resposta par, apresentado ao neurônio no momento t , categoricamente extraídos de um ambiente com ruído, cujas distribuições probabilísticas são desconhecidas.

$e_j(t) = d_j(t) - y_j(t)$, constitui o sinal de erro ora observado na saída do neurônio j no momento t .

O processo de aprendizado com supervisão tem como responsabilidade corrigir esta falha, onde são observado todos os neurônios da RNA, para isto busca minimizar um critério (função objetivo) baseado em $e_j(t)$, $j=1, 2, \dots, o$, onde o é o número de neurônios da RNA, de forma que, para t suficientemente alto, $y_j(t)$, $j=1, 2, \dots, o$, estejam mais próximos possível de $d_j(t)$, $j=1, 2, \dots, o$, no sentido estatístico como é ilustrado na Figura 11.



Rede multi-camadas Feedforward

Figura 11: Aprendizagem por correção de erro.

Fonte: Nascimento; Yoneyama (2004).

Uma forma muito utilizada é a de erro quadrático médio:

$$J = E \left[\frac{1}{o} \sum_{j=1}^o e_j^2(t) \right]$$

Em toda análise está implícito o conceito da hipótese de estacionariedade dos processos presentes e aleatórios. Portanto, para minimizar J é de suma importância conhecer as características probabilísticas do sistema.

A aproximação para este critério é usar o valor instantâneo do erro quadrático médio:

$$J \cong J(k) = \frac{1}{o} \sum_{j=1}^o e_j^2(k)$$

Nesta fase, vamos acreditar que a minimização de $J(t)$ é feita somente em relação aos pesos sinápticos da RNA.

As respostas obtidas pelas unidades são computadas por meio de uma função de ativação. Existem inúmeros tipos de funções de ativação, as mais comuns são: *Hard Limiter*, *Threshold Logic* e *Sigmóide*, como se observa na Figura 12.

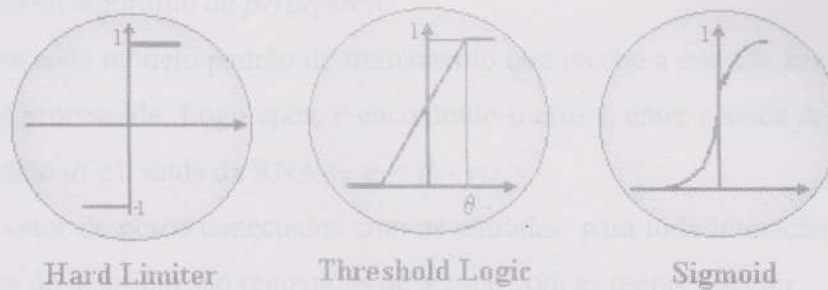


Figura 12: Funções de Ativação.
Fonte: Mendes (1998).

4.5.2 Tipos de Algoritmos de Aprendizado com Supervisão

4.5.2.1 O Perceptron

Rosenblatt (1962) avaliou o *perceptron* como a arquitetura mais básica de RNA capaz de classificar padrões linearmente separáveis. O primeiro modelo de treinamento supervisionado foi o *perceptron*, no entanto, alguns *perceptrons* fossem auto-organizados. Simplesmente, o *perceptron* baseia-se em uma única camada de neurônios com pesos sinápticos e bias (neurônio especial que serve para aumentar os graus de liberdade, permitindo uma melhor adaptação, por parte da rede neural, ao conhecimento à ela fornecido) configuráveis. Para que o algoritmo de treinamento do perceptron possua convergência garantida, os padrões de entrada precisam ser linearmente separáveis, ou seja, ser capaz de encontrar um conjunto de pesos que classifica de forma correta os dados. Os pesos dos neurônios, que constituem o perceptron, serão aqueles em que as superfícies de decisão realizadas pela RNA estavam apropriadamente posicionadas.

4.5.2.2 Perceptron Simples para Classificação de Padrões

Funcionamento do algoritmo do *perceptron*:

- Para cada modelo padrão de treinamento que recebe a entrada $\mathbf{x}_i$ e a saída da rede y_i é processada. Logo após, é encontrado o erro e_i entre a saída desejada para este padrão d_i e a saída da RNA y_i , $e_i = d_i - y_i$.
- O vetor de pesos conectados com as entradas, para toda interação, cada saída e o bias do neurônio são renovados de acordo com as regras a seguir:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + a e_i \mathbf{x}_i,$$

$$b(t+1) = t(t) + a e_i,$$

onde $\mathbf{w} \in \mathbb{R}^{1 \times m}$, $\mathbf{x} \in \mathbb{R}^{1 \times m}$ e $b \in \mathbb{R}^{1 \times m}$.

Considerando o caso mais básico de um *perceptron* apenas um neurônio como mostra a Figura 13.

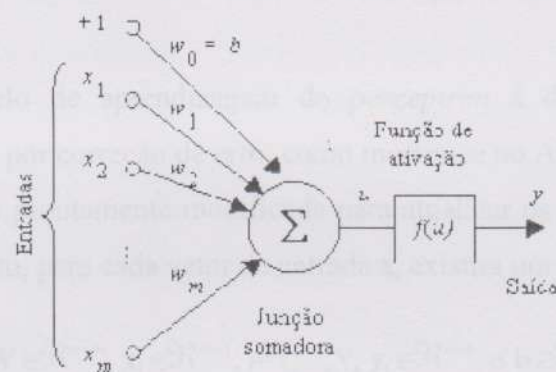


Figura 13: Modelo lógico genérico do neurônio.

Fonte: Nascimento; Yoneyama (2004).

O objetivo principal desta RNA, mais especificamente deste neurônio, é analisar e classificar padrões de entrada como pertencentes ou não, de uma dada classe. Considerando o conjunto de informações de entrada como sendo composto por N amostras $\{\mathbf{x}_1, d_1\}$, $\{\mathbf{x}_2, d_2\}$, ..., $\{\mathbf{x}_N, d_N\}$, onde $\mathbf{x}_j$ é o vetor j de entradas, e d_j sua saída desejada, onde esta é sua classe correspondente.

Seja $\mathbf{X} \in \mathbb{R}^{m \times N}$, a matriz de informações de entradas com N amostra de tamanho m cada, colunas de $\mathbf{X}$, e $\mathbf{d} \in \mathbb{R}^{1 \times N}$ o vetor de saídas esperadas. Observe abaixo o algoritmo 01, que pode ser utilizado para fazer o treinamento do *perceptron* de um único neurônio:

```

procedimento [w] = perceptron(max_it, E, α, X, d)
  atribuição inicial w //por simplicidade, atribui 0
  atribuição inicial b //por simplicidade, atribui 0
  t ← 1
  enquanto t < max_it & E > 0 faça,
    para i de 1 até N faça, //para cada padrão de entrada
       $y_i \leftarrow f(w x_i + b)$  //determine a saída para  $x_i$ 
       $e_i \leftarrow d_i - y_i$  //determine o erro para  $x_i$ 
       $w \leftarrow w + \alpha e_i x_i$  //atualize o vetor de pesos
       $b \leftarrow b + \alpha e_i$  //atualize o bias
    fim para
     $E \leftarrow \text{soma}(e_i)$ 
    t ← t + 1
  fim enquanto
fim procedimento

```

Algoritmo 01: Algoritmo *Perceptron* Simples.

Fonte: Neves (2004).

4.5.2.3 Perceptron de Múltiplos Neurônios

Veja que o modelo de aprendizagem do *perceptron* é do tipo supervisionada, utilizando a aprendizagem por correção de erro, como mostra-se no Algoritmo 02. Contudo, a regra em questão pode ser prontamente modificada para atualizar os pesos de uma RNA em uma única camada. Portanto, para cada vetor de entrada $\mathbf{x}_i$ existirá um vetor de saídas da rede:

$$\mathbf{y}_i = f(\mathbf{W}\mathbf{x}_i + \mathbf{b}); \mathbf{W} \in \mathbb{R}^{o \times m}, \mathbf{x}_i \in \mathbb{R}^{m \times 1}, i=1, \dots, N, \mathbf{y}_i \in \mathbb{R}^{o \times 1}, \text{ e } \mathbf{b} \in \mathbb{R}^{o \times 1}, \mathbf{D} \in \mathbb{R}^{o \times N}.$$

Temos agora um vetor de erros para cada amostra de entrada: $e_i = d_i - y_i$.

```

procedimento [W] = perceptron(max_it, α, X, D)
  atribuição inicial w //por simplicidade, atribui 0
  atribuição inicial b //por simplicidade, atribui 0
  t ← 1
  enquanto t < max_it faça,
    para i de 1 até N faça, //para cada padrão de entrada
       $y_i \leftarrow f(\mathbf{W}\mathbf{x}_i + \mathbf{b})$  //determine a saída da rede para  $x_i$ 
       $e_i \leftarrow d_i - y_i$  //determine o vetor de erros para  $x_i$ 
       $\mathbf{W} \leftarrow \mathbf{W} + \alpha e_i \mathbf{x}_i^T$  //atualize a matriz de pesos
       $\mathbf{b} \leftarrow \mathbf{b} + \alpha e_i$  //atualize o vetor de bias
    fim para
    t ← t + 1
  fim enquanto
fim procedimento

```

Algoritmo 02: Algoritmo de Aprendizagem com múltiplas Saídas.

Fonte: Neves (2004).

4.5.2.4 Características Básicas do Treinamento do Perceptron

Inicialmente, constatou-se que distintos conjuntos iniciais de pesos para o *perceptron* poderão levar a distintas superfícies de decisão.

- De fato, o problema de ajustamento supervisionado de pesos pode ser encarado como um processo de procura por um conjunto de pesos que melhoram uma certa superfície de erro.
- Contudo, uma escolha imprópria da situação inicial da RNA pode conduzir o algoritmo a uma convergência para excelente locais desta superfície de erro.

O Critério de convergência, em se tratando do *perceptron*, é provável garantir que, dado um grupo de padrões linearmente separáveis, o algoritmo tem a capacidade de localizar uma superfície de decisão capaz de classificar de forma correta os dados.

- Neste caso, é aceitável utilizar a determinação de erro igual a zero para todos as amostras de entrada como critério de convergência para o *perceptron* simples de categorização binária.
- Chamamos de épocas de treinamento, outro critério que pode ser aproveitado, onde o mesmo, é a adequação por uma quantidade finita de iterações.

O algoritmo de treinamento do *perceptron* possui basicamente o parâmetro de treinamento α que deve ser definido pelo usuário, como mostra-se abaixo no Algoritmo 03.

É de suma importância distinguir o processo de treinamento e aplicação da RNA.

- O treinamento da RNA corresponde ao processo de ajustamento de pesos.
- Depois de treinada, a RNA terá a capacidade de ser utilizada ao mesmo problema, de forma a constatar a qualidade do aprendizado, ou a outro problema, de forma a averiguar sua capacidade de generalização.

```

procedimento [y] = perceptron(W, b, Z)
  para i de 1 até N faça,      //para cada padrão de entrada
     $x_i$ 
     $y_i \leftarrow f(Wx_i + b)$     //determine as saídas da rede
  fim para
fim procedimento
  
```

Algoritmo 03: Algoritmo usado para treinar uma RNA *perceptron* uma camada.

Fonte: Neves (2004).

4.5.3 Aprendizagem sem Supervisão

No treinamento sem supervisão ou auto-organizado, não há um expectador externo para analisar o processo de aprendizagem (HAYKIN, 1999). Um Exemplo clássico de RNA com aprendizado sem supervisão são os Mapas de Auto-organização de Kohonen.

Os mapas de auto-organização, ou redes de Kohonen, foram pesquisados por Tuevo Kohonen, pesquisador da universidade de Helsinki – Europa. O modelo de Kohonen é uma RNA do tipo *feedforward*, de treinamento sem supervisão. É representada em um espaço bidimensional dividido em grupos de células, onde um desses grupos ou uma única célula pode responder a entrada na rede (STARKE, 1995).

Este tipo de RNA pode ser aplicada ao reconhecimento de padrões, robótica, controle de processos e processamento de informações semânticas, observe o modelo na Figura 14.

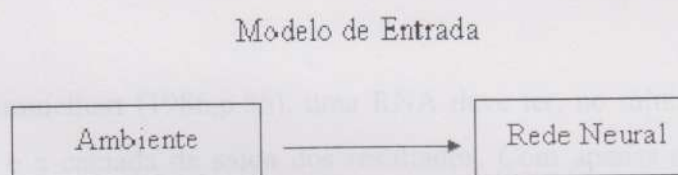


Figura 14: Aprendizagem auto-organizada.
Fonte: Mendes (1998).

Na maioria das vezes as RNA's auto-organizadas utilizam um **algoritmo competitivo de aprendizagem**. Com aprendizagem por competitividade, os neurônios de saída da rede concorrem entre si a todo instante, no intuito de se tornar ativos, onde no final haverá apenas um neurônio vencedor.

O algoritmo torna-se capaz de descobrir regularidades estatísticas em um conjunto de dados, a partir desta propriedade.

Os Neurônios individuais especializam-se em conjuntos ou grupos de padrões parecidos. Tornam-se detectores de características de diferentes classes de dados de entrada.

Formalização do processo de aprendizado competitivo:

Para que um determinado neurônio i vença, a distância entre este neurônio, entre o vetor de pesos $\mathbf{w}_i$ deste neurônio e um certo padrão de entrada $\mathbf{x}$, deve ser a menor entre todos os neurônios da RNA, especificada determinada uma métrica de distância $\|\cdot\|$ (na maioria das vezes utiliza-se a distância Euclidiana). A idéia central é encontrar o neurônio cujo vetor de pesos seja o mais parecido ao padrão da entrada, ou seja: $i = \arg \min_i \|\mathbf{x} - \mathbf{w}_i\|, \forall i$. Se um

neurônio não responde a um determinado padrão de entrada, ele não é o vencedor, portanto, nenhuma adaptação é feita neste neurônio.

No entanto, o neurônio i vencedor da competição sofre um determinado ajuste Δw_i no seu vetor de pesos, com direcionamento ao vetor de entrada:

$$\Delta w_i = \begin{cases} \alpha(x - w_i) & \text{se } i \text{ ganha a competição} \\ 0 & \text{se } i \text{ perde a competição} \end{cases}$$

onde α sinaliza o tamanho do passo a ser percorrido na direção de x . Este parâmetro α é chamado de taxa de aprendizagem.

4.6 TOPOLOGIAS DE REDES NEURAIS ARTIFICIAIS

Segundo Rummelhart (1986,p.86), uma RNA deve ter, no mínimo, duas camadas: a camada de entrada e a camada de saída dos resultados. Com apenas duas camadas, a rede tende apresentar um desempenho muito limitado, o acréscimo de uma camada intermediária faz-se necessária. Nesse tipo de configuração, cada neurônio está conectado com todos os outros das camadas vizinhas, porém os neurônios da mesma camada não se comunicam, além da comunicação ser unidirecional, apresentando, assim, um comportamento estático.

Para Kohonen (1972, p.357), com a presença de somente uma camada oculta (intermediária), já é possível calcular uma função arbitrária qualquer a partir de dados informados.

Independente da cada abordagem, quanto mais camadas de neurônios, melhor é o desempenho da RNA, pois potencializa a capacidade de aprendizado, aperfeiçoando a precisão com que a mesma demarca as regiões de decisão.

4.6.1 Redes Neurais Simétricas

São redes neurais cuja matriz (do grafo de conectividade) é simétrica, sendo um caso especial das RNA's com ciclos, como podemos observar na Figura 15.

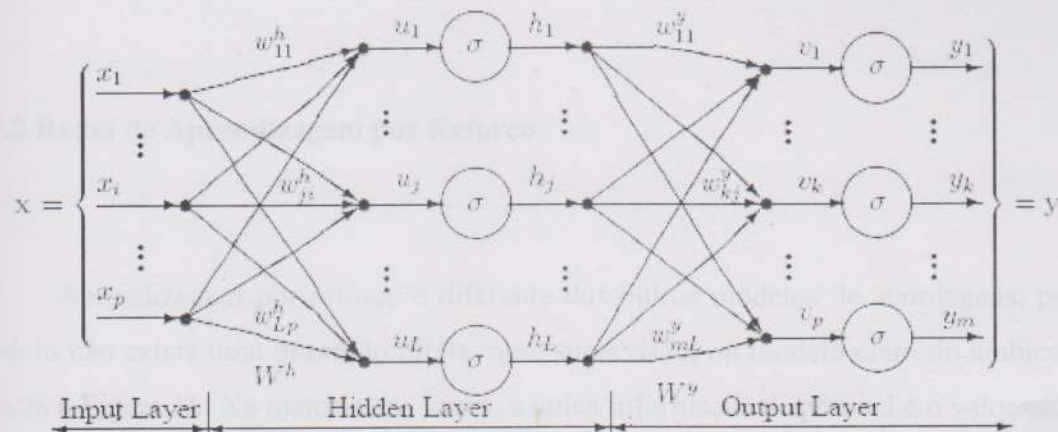


Figura 15: Rede Simétrica.

Fonte: Mendes (1998).

4.7 ALGUNS TIPOS DE REDES NEURAIS

4.7.1 Redes Neurais com Função Base Radial (RBF)

Segundo Haykin (1999,p 201), Broomhead e Lowe, em 1998, foram os primeiros a explorar o uso de RBF no projeto de RNA, as contribuições de Moody e Darken, Renal, em 1989, e Poggio e Girosi.

A construção de uma RNA RBF, em sua forma mais simples, envolve três camadas, cujos nodos de saída formam uma combinação linear das funções de base radial calculados pelos nodos da camada escondida.

As funções de base radial na camada escondida produzem uma resposta localizada para o estímulo (padrão) de entrada. Por essa razão, essa categoria de rede algumas vezes é referenciada na literatura como Redes de Campos Receptivos Localizados (TODESCO, 1995).

Uma RBF pode requerer mais neurônios do que Redes *Backpropagation*, mas, na maioria das vezes, elas são muito mais rápidas do que as Redes *FeedForward* padrões. Utilizando uma arquitetura de duas camadas, uma não linear e outra, geralmente linear, são conhecidas como excelentes aproximadores universais. Uma RNA RBF pode aproximar uma RNA *Perceptron* Multi-Camadas (MLP), entretanto, o contrário nem sempre é possível.

4.7.2 Redes de Aprendizagem por Reforço

Aprendizagem por reforço é diferente dos outros modelos de abordagens, pois neste modelo não existe uma interação direta, nem supervisão, ou modelo claro do ambiente como mostra a Figura 16. Na maioria das vezes, a única informação disponível é o valor escalar que aponta a qualidade do desenvolvimento da RNA. Neste modelo de aprendizagem existe um objetivo a ser alcançado. Durante o processo de aprendizagem, a RNA faz tentativas de ações de saídas e recebe um sinal de reforço do ambiente que possibilita medir a qualidade de sua ação. O sistema de aprendizagem retém seletivamente as ações que levam os sinais de reforço a uma maximização, a fim de potencializar ao máximo a taxa de aprendizado.

A cada iteração t , o sistema de aprendizagem recebe uma nova entrada $\mathbf{x}(t)$, que representa o estado do ambiente e fornece uma saída $y(t)$, e no passo seguinte recebe um escalar de reforço representado por $r(t+1)$, e um novo estado do ambiente representado por $\mathbf{x}(t+1)$.

Desta forma, os dois conceitos genéricos por trás do modelo de aprendizagem por reforço são: **Busca por tentativa e erro** e **Reforço retardado**.

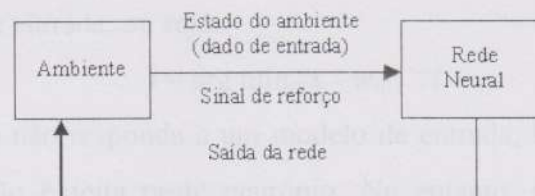


Figura 16: Aprendizagem por reforço.
Fonte: Neves (2004).

4.7.3 RNA com Competitividade

Uma RNA competitiva básica é do modelo *feedforward*, de apenas uma camada. Os neurônios de saída são lineares e iguais, com exceção dos seus relativos vetores de pesos associados. Desta forma, para cada padrão de entrada apresentado os neurônios de saída irão concorrer entre si a fim de determinar aquele neurônio que é mais estimulado pelo padrão de entrada, com um único neurônio sendo ativado para cada padrão. Esta especialidade é chamada de *winner-takes-all*. O neurônio vitorioso da competição será levado em direção do padrão de entrada, como ilustrado na Figura 17 abaixo.

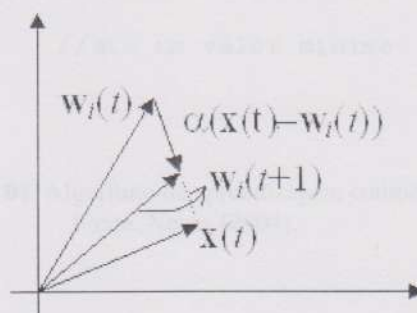


Figura 17: Representação de competitividade entre neurônios.
Fonte: Nascimento; Yoneyama (2004).

Formalmente, os neurônios da RNA aprendem e se especializam em grupos de padrões similares chamados de *clusters* que se tornam detectores ou extratores de características para grupos distintos de padrões de entrada. Para que o vencedor seja o neurônio i , a distância do vetor de pesos $\mathbf{w}_i$ deste neurônio e um apontado padrão de entrada $\mathbf{x}$ deve ser a mínima em meio a todos os demais neurônios da RNA, dada uma métrica de deformidade ou de abscissa $\|\cdot\|$. O objetivo é achar o neurônio cujo vetor de pesos seja o mais semelhante à amostra de entrada, ou seja:

$$i = \arg \min_i \|\mathbf{x} - \mathbf{w}_i\|; \forall i$$

Caso o neurônio não responda a um modelo de entrada, logo, este não é o vitorioso, então nenhuma alteração é feita neste neurônio. No entanto, o neurônio i que venceu a competição sofre uma alteração $\Delta \mathbf{w}_i$ no seu vetor de pesos em direção do vetor de entrada:

$$\Delta \mathbf{w}_i = \begin{cases} \alpha(\mathbf{x} - \mathbf{w}_i) & \text{se } i \text{ ganha a competição} \\ 0 & \text{se } i \text{ perde a competição} \end{cases}$$

Onde “ α ” contém a dimensão do passo a ser dado em direção a $\mathbf{x}$. O parâmetro “ α ” é chamado de taxa de aprendizagem. Na maioria das vezes a taxa de aprendizagem é

minimizada praticamente a zero no decorrer do processo de adaptação, o que garante uma estabilidade do método auto-organizado no decorrer do tempo. A sequência de aprendizagem de uma RNA competitiva está sucintamente no Algoritmo 04.

```

procedimento [W] = compet(max_it,  $\alpha_0$ , X)
  atribui valores iniciais W //normalmente uma
  distribuição aleatória
  t  $\leftarrow$  1
  enquanto t < max_it faça,
    vet_permut  $\leftarrow$  ale_perm(N) //permutações de N
    para j de 1 até N faça, //para cada padrão de
    entrada
      //selecione o padrão i a ser apresentado
      i  $\leftarrow$  vet_permut(j) //apresentação aleatória
       $i(x) \leftarrow \arg \min_j \|x_i - w_j(t)\|$ ,  $j = 1, \dots, o$  //competição
       $w_i(t+1) = w_i(t) + \alpha(t) [x(t) - w_i(t)]$  //cooperação e adaptação
    fim para
     $\alpha \leftarrow \text{reduz}(\alpha)$  //até um valor mínimo
    t  $\leftarrow$  t + 1
  fim enquanto
fim procedimento

```

Algoritmo 04: Algoritmo de aprendizagem competitiva.
Fonte: Neves (2004).

4.7.4 Redes Neurais Recorrentes

As RNA's recorrentes também são conhecidas como RNA de Elman que é composta por duas camadas explícitas e uma escondida, com *feedback* na primeira camada. Essa conexão de recorrência permite que este tipo de rede seja utilizada tanto para detectar quanto para gerar padrões variando no tempo. A RNA de Elman possui neurônios *tansig* (*Hyperbolic tangent sigmoid transfer function*) na camada escondida (recorrente) e neurônios *purelin* (*linear transfer function*) na camada de saída, essa combinação pode aproximar qualquer função, com o número finito de descontinuidades, com precisão arbitrária. O único requerimento é que deve haver um número de neurônios suficiente na camada escondida (HAYKIN, 1999).

As RNA's recorrentes são um dos tipos principais de arquitetura de RNA's, pois estas possuem, ao menos, um laço de petição que realimenta a saída de neurônios para os outros neurônios da RNA como se observa na Figura 18.

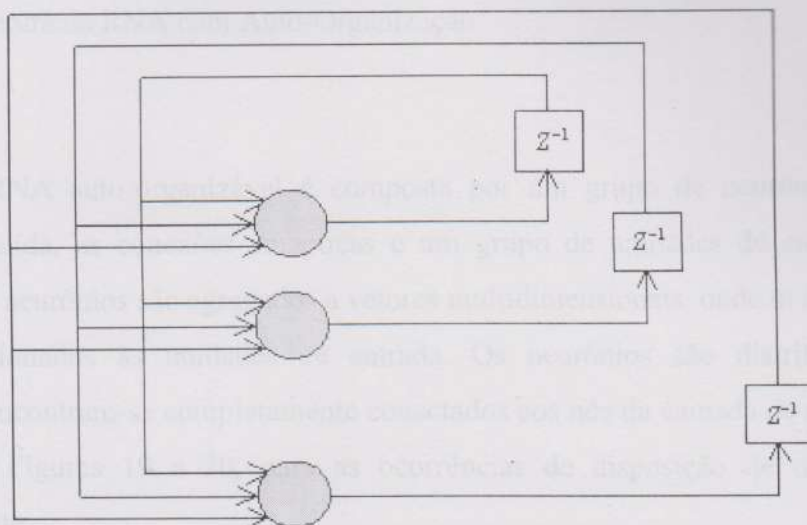


Figura 18: RNA neural recorrente.
Fonte: Nascimento; Yoneyama (2004).

4.7.5 RNA com Auto-Organização

Também conhecida como *Self-Organizing* é um dos tópicos mais interessantes no campo de RNA's. Tais redes podem aprender a detectar regularidades e correlações em sua entrada e adaptar sua resposta futura em conformidade com a entrada. Os neurônios da rede competitiva aprendem a reconhecer grupo de vetores de entrada similares. *Self-Organizing Maps* (SOM) aprendem a reconhecer grupos de vetores de entrada similares de maneira que os neurônios fisicamente próximos na camada de neurônios respondam de forma similar a um vetor de entrada (HAYKIN, 1999).

No geral, os mecanismos constituídos pelos componentes de certo sistema apto a propagar a auto-organização são: realimentação negativa, realimentação positiva e interação local.

4.7.5.1 Arquitetura de RNA com Auto-Organização

Uma RNA auto-organizável é composta por um grupo de neurônios, que são as unidades de saída, as conexões sinápticas e um grupo de unidades de entrada. Os pesos sinápticos dos neurônios são agregados a vetores multidimensionais, onde m é a dimensão dos padrões relacionados às unidades de entrada. Os neurônios são distribuídos em uma disposição e encontram-se completamente conectados aos nós da camada de entrada, segundo ilustrado nas Figuras 19 e 20, para as ocorrências de disposição de dimensão 1 e 2, respectivamente.

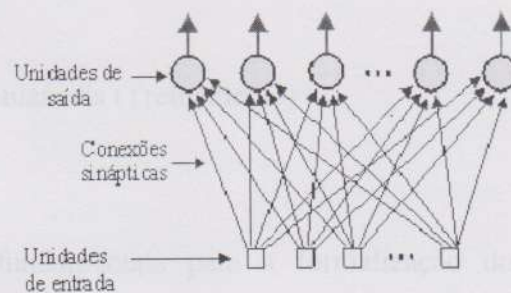


Figura 19: Arquitetura típica de uma RNA auto-organizável unidimensional.
Fonte: Nascimento; Yoneyama (2004).

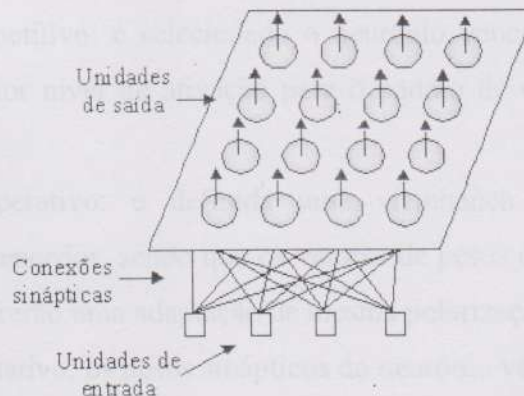


Figura 20: Arquitetura típica de uma RNA auto-organizável bidimensional.
Fonte: Nascimento; Yoneyama (2004).

A Figura 20 mostra uma possibilidade de distribuição de pesos sinápticos para uma representação auto-organizável com $m = 2$ e dimensão 1. Observe que, mesmo que a disposição se modele com o formato que permita os neurônios se arrumarem livremente no espaço original dos dados, sua dimensão prossegue sendo unitária. Desta forma, a explicação da disposição, depois do término do processo de auto-organização dos neurônios, será em um

espaço unidimensional, independentemente da dimensão original dos dados, como mostra a Figura 21.

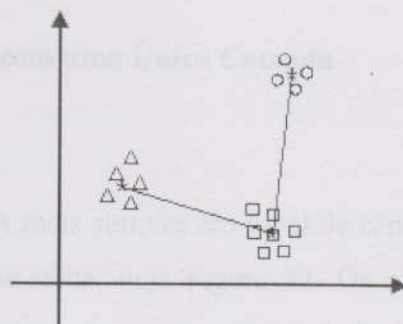


Figura 21: Possível distribuição de pesos para um SOM unidimensional.
Fonte: Nascimento; Yoneyama (2004).

4.7.5.2 Mapas auto-organizáveis (Treinamento)

Quatro etapas fundamentais para a formalização do algoritmo de mapas auto-organizáveis:

- Inicialização sináptica: os vetores de pesos sinápticos iniciais dos neurônios presentes na rede são definidos de acordo com algum critério pré-estabelecido.
- Processo competitivo: é selecionado o neurônio vencedor. Este será aquele que apresentar maior nível de ativação para o padrão de entrada apresentado à rede neural.
- Processo cooperativo: é definida uma vizinhança topológica de neurônios associada ao vencedor, sendo que os vetores de pesos de todos os neurônios desta vizinhança sofrerão uma adaptação de mesma polarização.
- Processo adaptativo: os pesos sinápticos do neurônio vencedor e de sua vizinhança são modificados.

Geralmente, o treinamento é complementado quando o mapa alcança um estado de equilíbrio, em que não incidi alterações expressivas nos pesos sinápticos, após a exposição de todos os padrões de treinamento.

4.8 REDES FEEDFORWARD

4.8.1 Rede Feedforward com uma Única Camada

É o caso de RNA mais simples em nível de camadas, pois consiste numa camada de entrada e uma camada de saída, veja Figura 22. Os neurônios de entrada geralmente são lineares, ou seja, simplesmente propagam o sinal de entrada até a próxima camada. Conhecidos também como neurônios sensoriais.

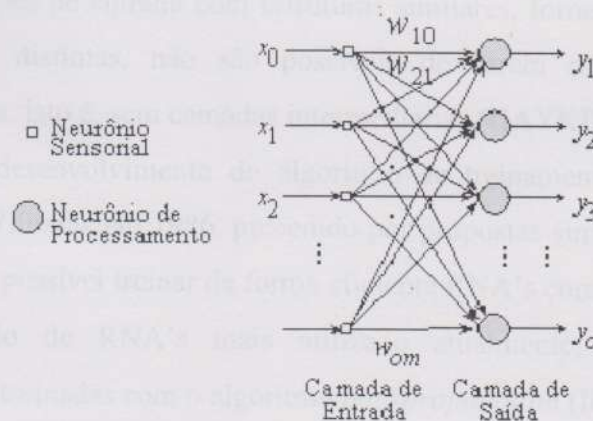


Figura 22: RNA feedforward com uma única camada.
Fonte: Mendes (1998)

Esta RNA é chamada de *feedforward*, porque a propagação do sinal ocorre apenas em um sentido, da entrada para a saída, portanto, é apenas no sentido positivo, como mostra a Figura 23.

$$\mathbf{W} = \begin{bmatrix} w_{10} & w_{11} & \cdots & w_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{o0} & w_{o1} & \cdots & w_{om} \end{bmatrix}$$

Figura 23: Matriz $\mathbf{W}$ (propagação).
Fonte: Neves (2004).

Note que a primeira coluna de $\mathbf{W}$ corresponde ao vetor de bias. Em forma matricial:

$$\mathbf{y} = f(\mathbf{W} \cdot \mathbf{x}), \text{ onde } \mathbf{W} \in \mathbb{R}^{o \times m}, w_i \in \mathbb{R}^{1 \times m}, i = 1, \dots, o, \mathbf{x} \in \mathbb{R}^{m \times 1}, \text{ e } \mathbf{y} \in \mathbb{R}^{o \times 1}$$

4.8.2 - Rede *Feedforward* de Múltiplas Camadas (MPL)

Quando as RNA's de uma só camada são utilizadas, os padrões de treinamento apresentados a entrada são mapeados inteiramente em um conjunto de padrões de saída da rede, ou seja, não é possível a formação de uma representação interna. Neste caso, a codificação proveniente do mundo exterior deve ser suficiente para implementar esse mapeamento (FAUSETT, 1994).

Esta restrição implica que padrões de entradas parecidos resultem em muitos padrões de saída similares, o que leva o sistema à inaptidão de aprender importantes mapeamentos. Como resultado, padrões de entrada com estruturas similares, fornecidos do mundo externo, que levam a saídas distintas, não são possíveis de serem mapeadas por redes sem representações internas, isto é, sem camadas intermediárias (HAYKIN, 1999).

Entretanto, o desenvolvimento do algoritmo de treinamento *backpropagation*, por Rumelhart, Hilton e Williams em 1986, precedido por propostas similares ocorridas nos anos 70 e 80, indicou que é possível treinar de forma eficiente RNA's com camadas intermediárias, resultando no modelo de RNA's mais utilizado atualmente, as RNA's *Perceptron* Multicamadas (MPL), treinadas com o algoritmo *backpropagation* (Ibidem).

Denominadas “**redes de múltiplas camadas**”, porque possuem uma ou mais camadas intermediárias ou camadas escondidas. Ao adicionar camadas intermediárias, não-lineares, possibilita-se aumentar o poder de processamento de uma RNA *feedforward*. A saída de cada camada é basicamente utilizada como alimentação da entrada para a camada seguinte. Geralmente o algoritmo de treinamento neste tipo de RNA, utiliza a retropropagação do erro entre a saída da rede e uma saída desejada que por sua vez já seja conhecida, como podemos observar na Figura 24.

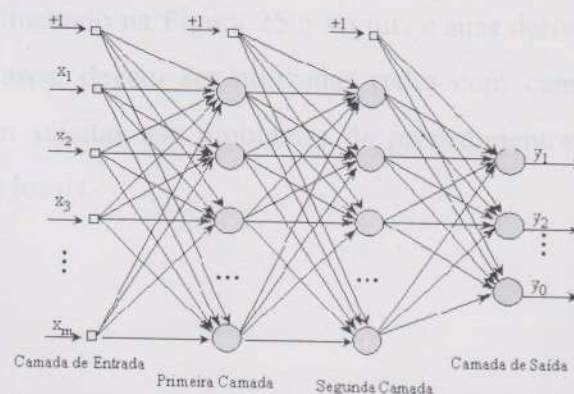


Figura 24: RNA feedforward de múltiplas camadas.
Fonte: Neves (2004).

Seja $\mathbf{W}^k$ a matriz de pesos da camada k , contadas da esquerda para a direita. w_{ij}^k correspondente ao peso ligando o neurônio pós-sináptico i ao neurônio pré-sináptico j na camada k . Na conjectura matricial, a saída da rede é dada por:

$$\mathbf{y} = \mathbf{f}^3(\mathbf{W}^3 \mathbf{f}^2(\mathbf{W}^2 \mathbf{f}^1(\mathbf{W}^1 \mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3)$$

Note que $\mathbf{f}^k$, $k = 1, \dots, M$ onde (M = número de camadas da rede) é uma matriz quadrada $\mathbf{f}^k \in \mathbb{R}^{N \times N}$, onde N é a quantidade numérica de neurônios na camada k .

4.9 O ALGORITMO DE BACKPROPAGATION

Durante o treinamento com o algoritmo *backpropagation*, a rede opera em uma seqüência de dois passos. Primeiro, um padrão é apresentado à camada de entrada da rede. A atividade resultante flui através da rede, camada por camada, até que a resposta seja produzida pela camada de saída. No segundo passo, a saída obtida é comparada a desejada para este padrão particular. Se não estiver correta, o erro é calculado. O erro é propagado a partir da camada de saída até a camada de entrada, e os pesos das conexões das unidades das camadas internas são modificados conforme o erro é retropropagado (ACKLEY, 1985). As redes que utilizam *backpropagation* trabalham com uma variação da regra delta apropriada para redes multicamadas: a regra delta generalizada. A regra delta padrão essencialmente implementa um gradiente descendente no quadrado da soma do erro para funções de ativação lineares. Redes sem camadas intermediárias podem resolver problemas onde a superfície de erro tem a forma de parabolóide com apenas um mínimo. Entretanto, a superfície do erro pode não ser tão simples, como é ilustrado na Figura 25 a seguir, e suas derivadas mais difíceis de serem calculadas. Nesses casos, devem ser utilizadas redes com camadas intermediárias. Ainda assim, as redes ficam sujeitas aos problemas de procedimentos *Hill-climbing*, ou seja, ao problema de mínimos locais.

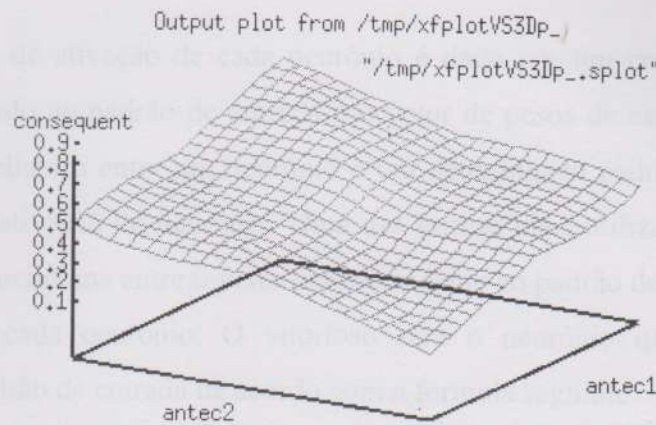


Figura 25: Exemplo de superfície de erro.
Fonte: Mendes (1998).

A taxa de aprendizado é uma constante de proporcionalidade no intervalo $[0,1]$, pois este procedimento de aprendizado requer apenas que a mudança no peso seja proporcional à reta (HAYKIN, 1999).

Enfim, o treinamento das redes MPL com *backpropagation* pode demandar muitos passos no conjunto de treinamento, o resultando um tempo de treinamento consideravelmente longo. Se for encontrado um mínimo local, o erro para o conjunto de treinamento pára de diminuir e estaciona em um valor maior que o aceitável. Uma maneira de aumentar a taxa de aprendizado sem levar à oscilação é modificar a regra delta generalizada para incluir o termo *momentun*, uma constante que determina o efeito das mudanças passadas dos pesos na direção atual do movimento no espaço de pesos (FAUSETT, 1994).

4.10.2 Processo Cooperativo

4.10 PROCESSOS EM RNA'S

4.10.1 Processo Competitivo

Durante o processo de competitividade, os vetores relacionados aos padrões de entrada são consecutivamente apresentados ao mapa auto-organizável, como se observa na Figura 26. O mesmo padrão de entrada é dado para todos os neurônios que em seguida competem entre

si para serem ativados, onde apenas um neurônio será o vencedor baseado no padrão estabelecido. Essa norma de competição é chamada de "o vencedor leva tudo" (*winner-takes-all*).

O coeficiente de ativação de cada neurônio é dado por um critério de semelhança entre o vetor apropriado ao padrão de entrada e o vetor de pesos de cada neurônio. Quanto maior o grau de semelhança entre um neurônio e um determinado padrão de entrada, maior será o coeficiente de ativação do neurônio. Uma das formas mais utilizadas na literatura é o inverso da distância euclidiana entre o vetor correspondente ao padrão de entrada e o vetor de pesos sinápticos de cada neurônio. O vitorioso será o neurônio que apresentar maior semelhança com o padrão de entrada de acordo com a fórmula seguinte:

$$J = \arg \min_j \| \mathbf{x} - \mathbf{w}_j \|$$

Onde $\mathbf{w}_j$ é o vetor de pesos do neurônio j , $\mathbf{x}$ é o vetor correspondente ao padrão de entrada e $\|.\|$ é a norma Euclidiana.

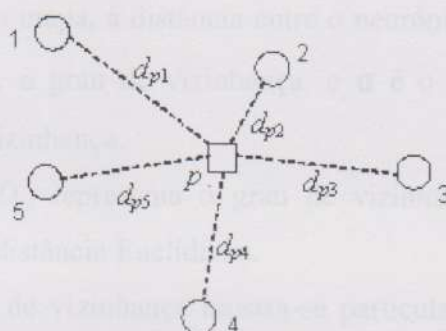


Figura 26: Processo competitivo: seleção do neurônio vencedor.

Fonte: Nascimento; Yoneyama (2004).

4.10.2 Processo Cooperativo

O método cooperativo é fundamentado em uma estrutura neurobiológica, no qual um neurônio que se encontra na condição "ativa" tende a influenciar o estado dos neurônios mais próximos, de modo que vizinhos situados topologicamente mais próximos sofrem maior influência. Os tipos de vizinhança a serem considerados aqui envolvem 1 e 2 dimensões. A Figura 27 ilustra relações de vizinhança unidimensional e bidimensional.

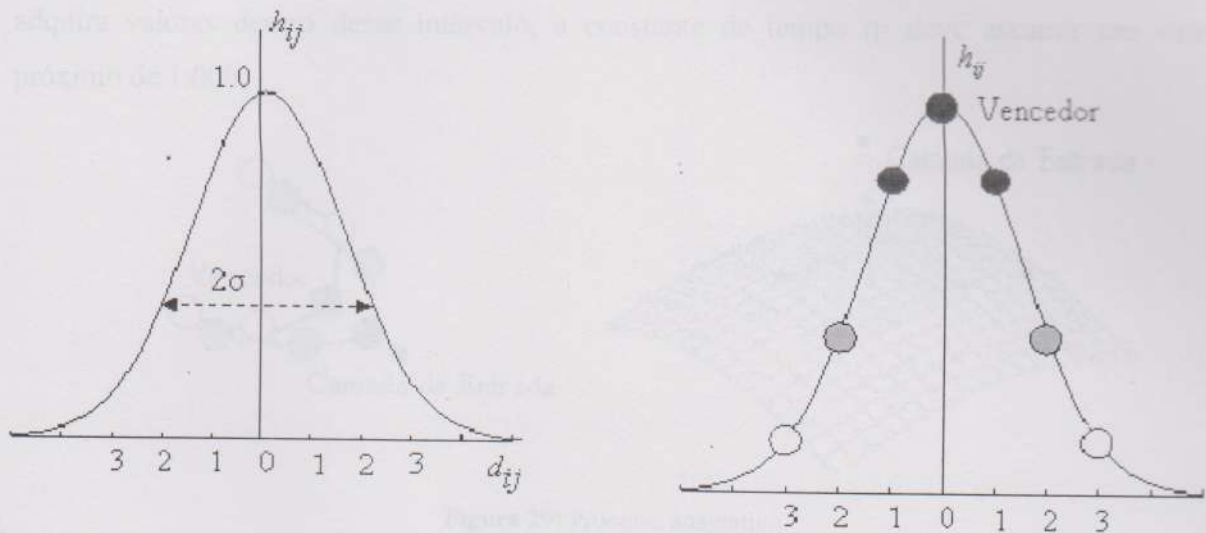


Figura 28: Processo cooperativo: relações de vizinhança gaussiana.
Fonte: (NASCIMENTO; YONEYAMA, 2004).

4.10.3 Processo Adaptativo

No processo adaptativo, os vetores de pesos sinápticos do neurônio vitorioso e de sua vizinhança são alterados de forma que ele se aproxime do padrão de entrada apresentado, respeitando a relação de vizinhança determinada durante o processo cooperativo, como observa-se adiante na Figura 29 e posteriormente no Algoritmo 05. A atualização do vetor de pesos sinápticos $\mathbf{w}_j$ é dada por:

$$\mathbf{w}_j = \mathbf{w}_j + a\mathbf{w}_j$$

Onde $h_{j,i}(\mathbf{x})$ é a função de ativação de vizinhança definida anteriormente, $\mathbf{x}$ é o vetor associado ao padrão de entrada, $\eta(n)$ é a taxa de aprendizagem na época n , e $\mathbf{w}_j$ é o vetor de pesos sinápticos associado ao neurônio j .

A atualização da taxa de aprendizagem é dada pela seguinte equação:

$$\eta(n) = \eta_0 \exp\left(-\frac{n}{\tau_2}\right)$$

Onde n_2 é uma constante de tempo e n_0 é a taxa de aprendizagem inicial. Basicamente, a taxa de aprendizado inicial n_0 assume um valor próximo de 0,1 e deve decrescer pouco a pouco até um valor próximo de 0,01. Para que a taxa de aprendizagem

adquira valores dentro desse intervalo, a constante de tempo η_2 deve assumir um valor próximo de 1.000.

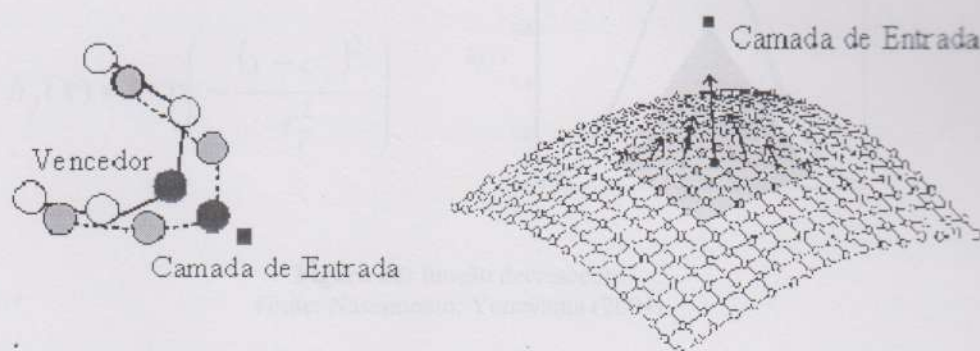


Figura 29: Processo adaptativo.

Fonte: Nascimento; Yoneyama (2004).

```

procedimento [W] = som(max_it,  $\alpha_0$ ,  $C_0$ ,  $\tau_1$ ,  $\tau_2$ , X)
  atribuição inicial W //valores pequenos aleatórios ( $w_i \neq w_j$ 
  para  $i \neq j$ )
  t  $\leftarrow$  1
  enquanto t < max_it faça,
    vet_permut  $\leftarrow$  aleatoriapermut(N) //permutação de N
    para j de 1 até N faça, //para todos padrões de entrada
      //selecione o índice i do padrão  $x_i$  para ser apresentado
      i  $\leftarrow$  vet_permut(j) //apresentar padrões aleatoriamente
      //competição e determinação do vencedor i(x)
       $i(x) \leftarrow \arg \min_j \|x_i - w_j(t)\|$ ,  $j = 1, \dots, P$ 
      //cooperação e adaptação
       $w_i(t+1) = w_i(t) + \alpha(t) h_{ji}(x) [x(t) - w_i(t)]$ 
      //atualiza  $\alpha$  e a função da vizinhança
    fim para
     $\alpha \leftarrow \text{reduce}(\alpha)$  //até  $\alpha$  encontrar um valor mínimo
     $\sigma \leftarrow \text{reduce}(\sigma)$  //até  $h_{ji}(x) = 1$ 
    t  $\leftarrow$  t + 1
  fim enquanto
fim procedimento
  
```

Algoritmo 05: Algoritmo para o padrão Kohonen mapa auto-organizado.
Fonte: Neves (2004).

4.11 FUNCIONAMENTO DE REDES NEURAIS ARTIFICIAIS RBF

Uma função de ativação de base radial é caracterizada por apresentar uma resposta que decresce (ou cresce) monotonicamente com a distância a um ponto central.

O centro e a taxa de decrescimento (ou crescimento) em cada direção são alguns dos parâmetros a serem definidos. Estes parâmetros devem ser constantes caso o modelo de regressão seja tomado como linear nos parâmetros ajustáveis.

Uma função de base radial monotonicamente decrescente típica, como podemos observar na Figura 31, é a função gaussiana, dada na forma:

$$\square \quad h_j(x) = \exp\left(-\frac{(x - c_j)^2}{r_j^2}\right)$$

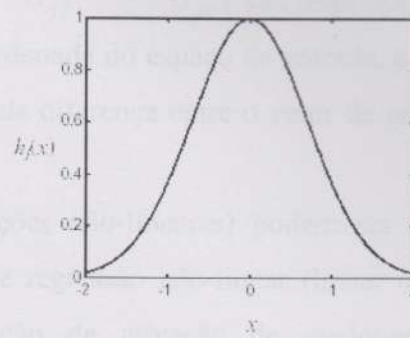


Figura 30: função decrescente.
Fonte: Nascimento; Yoneyama (2004).

Uma função de base radial monotonicamente crescente típica, como podemos observar na Figura 31, é a função multiquádrica dada na forma:

$$\square \quad h_j(x) = \frac{\sqrt{r_j^2 + (x - c_j)^2}}{r_j}$$

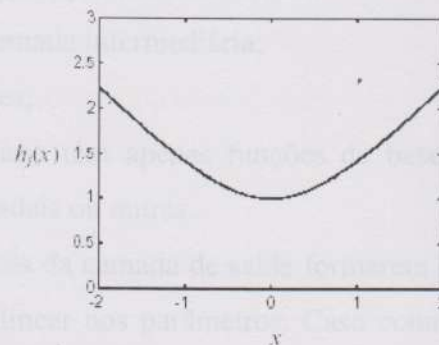


Figura 31: função crescente.
Fonte: Nascimento; Yoneyama (2004).

No caso multidimensional e tomando a função gaussiana, $h_j(\mathbf{x})$ assume a forma:

$$h_j(\mathbf{x}) = \exp\left(-(\mathbf{x} - \mathbf{c}_j)^T \Sigma_j^{-1} (\mathbf{x} - \mathbf{c}_j)\right)$$

onde $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]^T$ é o vetor de entradas, $\mathbf{c}_j = [c_{j1} \quad c_{j2} \quad \cdots \quad c_{jn}]^T$ é o vetor que define o centro da função de base radial e a matriz Σ_j é definida positiva e diagonal, dada por:

$$\Sigma_j = \begin{bmatrix} \sigma_{j1} & 0 & \cdots & 0 \\ 0 & \sigma_{j2} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_{jn} \end{bmatrix},$$

de modo que $h_j(\mathbf{x})$ pode ser expandida na forma:

$$h_j(\mathbf{x}) = \exp\left(-\frac{(x_1 - c_{j1})^2}{\sigma_{j1}} - \frac{(x_2 - c_{j2})^2}{\sigma_{j2}} - \cdots - \frac{(x_n - c_{jn})^2}{\sigma_{jn}}\right)$$

Neste caso, os elementos do vetor $\sigma_j = [\sigma_{j1} \quad \sigma_{j2} \quad \dots \quad \sigma_{jn}]^T$ são responsáveis pela taxa de decrescimento da gaussiana junto a cada coordenada do espaço de entrada, e o argumento da função exponencial é uma norma ponderada da diferença entre o vetor de entrada e o centro da função de base radial.

As funções de base radial (são funções não-lineares) podem ser utilizadas como funções-base em qualquer tipo de modelo de regressão não-linear (linear ou não-linear nos parâmetros) e, particularmente, como função de ativação de qualquer tipo de rede multicamada. O fato do modelo de regressão resultante ser linear ou não-linear nos parâmetros se deve à possibilidade ou não de se ajustar os centros e as dispersões das funções. As redes neurais com funções de ativação de base radial (RBF) apresentam três diferenças principais em relação às redes tipo perceptron multicamadas:

- Elas sempre apresentam uma única camada intermediária;
- Neurônios de saída são sempre lineares;
- Os neurônios da camada intermediária têm apenas funções de base radial como função de ativação, ao invés de funções sigmoidais ou outras.

Como exposto acima, se apenas os pesos da camada de saída formarem o conjunto de parâmetros ajustáveis, então a rede neural é linear nos parâmetros. Caso contrário, ou seja, quando os centros $\mathbf{c}_j$ e as matrizes $\Sigma_j, j = 1, \dots, n$, também são ajustáveis, a rede neural é não-linear nos parâmetros, admitindo o próprio algoritmo de retro-propagação do erro para o processo de ajuste via treinamento supervisionado, como feito no caso do perceptron multicamadas.

A arquitetura da rede é apresentada na Figura 32, para o caso de uma única saída, resultando no seguinte mapeamento de entrada-saída:

$$y = \sum_{j=1}^m w_j h_j(\mathbf{x})$$

Caso $\mathbf{c}_j$ e $\Sigma_j, j = 1, \dots, n$, sejam ajustáveis, a saída assume a forma:

$$y = \sum_{j=1}^m w_j h_j(\mathbf{c}_j, \Sigma_j, \mathbf{x}).$$

Substituindo as formas compactas e expandidas de $h_j(\mathbf{x})$, dadas respectivamente pelas equações, resultam:

$$y = \sum_{j=1}^m w_j \exp(-(x - c_j)^T \Sigma_j^{-1} (x - c_j))$$

$$y = \sum_{j=1}^m w_j \exp\left(-\frac{(x_1 - c_{j1})^2}{\sigma_{j1}} - \frac{(x_2 - c_{j2})^2}{\sigma_{j2}} - \dots - \frac{(x_n - c_{jn})^2}{\sigma_{jn}}\right)$$

uma versão para múltiplas saídas é apresentada na Figura 33.

A consequência imediata do uso de funções de ativação de base radial está na forma como as entradas são processadas pelos neurônios da camada intermediária. Ao invés da ativação interna de cada neurônio da camada intermediária se dar pelo emprego do produto escalar entre o vetor de entradas e o vetor de pesos, como no caso do perceptron, ela é obtida a partir de uma norma ponderada da diferença entre ambos os vetores.

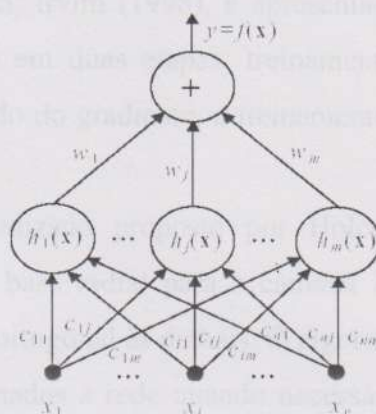


Figura 32: rede neural artificial de base radial.

Fonte: Nascimento; Yoneyama (2004).

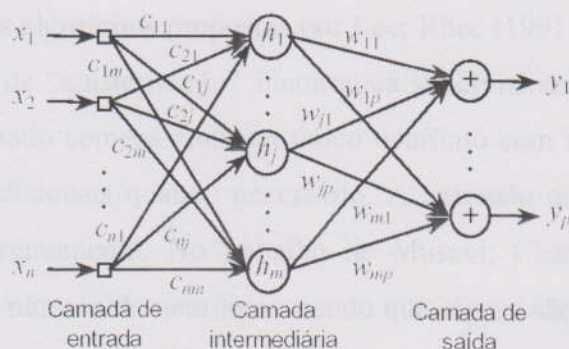


Figura 33: rede neural artificial de base radial com múltiplas saídas.

Fonte: Nascimento; Yoneyama (2004).

4.11.1 Métodos de Treinamento

Várias abordagens para o treinamento de redes neurais com funções de base radial têm sido propostas. Geralmente, elas podem ser divididas em duas partes:

- Definição dos centros, forma e dispersão das funções de base radial, normalmente baseada em treinamento não-supervisionado (quantização vetorial ou algoritmo de treinamento competitivo) ou computação evolutiva;
- Aprendizado dos pesos da camada de saída, responsáveis pela combinação linear das ativações da camada intermediária, empregando técnicas como pseudo-inversão e *Orthogonal Least Squares* (OLS) (CHEN; CHEN; CHANG, 1991).

Em Billings; Zheng (1995) é proposto um algoritmo genético para evoluir somente os centros, sendo que estes assumem valores inteiros distintos. Após determinados os centros das funções, os pesos são encontrados utilizando o algoritmo OLS.

Em Mocloone; Bronwn; Irvim (1998), é apresentada uma estratégia híbrida, a qual divide o treinamento da rede em duas etapas: treinamento da parte não-linear (centros e variâncias), utilizando o método do gradiente, e treinamento da parte linear (pesos de saída), utilizando OLS.

O algoritmo de aprendizado proposto por Holcomb & Morari (1992) introduz incrementalmente funções de base radial para a camada intermediária, de modo que cada função seja aproximadamente ortogonal às demais. O algoritmo inicia com um único neurônio e outros neurônios são adicionados à rede quando necessário. A localização dos centros da função de base radial associada a cada neurônio é otimizada usando técnicas convencionais de otimização. Entretanto, o critério de término deste algoritmo construtivo não é robusto e pode ser dependente do problema em questão (BILLINGS; ZHENG, 1995).

Já em relação aos algoritmos propostos por Lee; Rhee (1991) e Musavi; Chan (1992), são utilizados métodos de "clusterização" hierárquica supervisionada. No trabalho de Lee; Rhee (1991), o aprendizado começa com um único neurônio com uma base radial ampla, e introduz-se neurônios adicionais quando necessário. A extensão das bases radiais pode ser alterada ao longo do treinamento. No trabalho de Musavi; Chan (1992), o aprendizado começa com um grande número de neurônios, sendo que alguns são combinados (resultando em um único) quando possível, atualizando os centros e bases a partir daquelas previamente existentes antes da combinação. Estes dois algoritmos foram primeiramente desenvolvidos

para reconhecimento de padrões e podem não fornecer soluções adequadas para problemas de identificação de sistemas (BILLINGS; ZHENG, 1995).

4.11.2 Capacidade de Aproximação Universal

Dado um número suficiente de neurônios com função de base radial, qualquer função contínua definida numa região compacta pode ser devidamente aproximada usando uma RNA RBF (BILLINGS; ZHENG, 1995).

As redes RNA são redes de aprendizado local, de modo que é possível chegar a uma boa aproximação desde que um número suficiente de dados para treinamento seja fornecido na região de interesse.

Em contraste, *perceptrons* multicamadas são redes de aprendizado “global” (em virtude da natureza das funções de ativação) que fazem aproximações de efeito global, em regiões compactas do espaço de aproximação.

Redes neurais com capacidade de aproximação local são muito eficientes quando a dimensão do vetor de entradas é pequena. Entretanto, quando o número de entradas não é pequeno, as redes *MLP* apresentam uma maior capacidade de generalização (HAYKIN, 1999).

Isto ocorre porque o número de funções de base radial deve aumentar exponencialmente com o aumento da dimensão da entrada.

4.11.3 O Método dos Quadrados Mínimos para Modelos Lineares

Quando o treinamento supervisionado é aplicado a modelos lineares nos parâmetros, o método dos quadrados mínimos conduz a um problema de otimização que apresenta solução na forma fechada.

Assim, com um modelo de regressão linear na forma (assumindo um única saída):

$$f(\mathbf{x}) = \sum_{j=1}^m w_j h_j(\mathbf{x})$$

e o conjunto de treinamento dado $\{(\mathbf{x}_i, s_i)\}_{i=1}^N$ por, o método dos quadrados mínimos se ocupa em minimizar (em relação aos coeficientes da combinação linear) a soma dos quadrados dos erros produzidos a partir de cada um dos N padrões de entrada-saída.

$$\min_{\mathbf{w}} J(\mathbf{w}) = \min_{\mathbf{w}} \sum_{i=1}^N (s_i - f(\mathbf{x}_i))^2 = \min_{\mathbf{w}} \sum_{i=1}^N \left(s_i - \sum_{j=1}^m w_j h_j(\mathbf{x}_i) \right)^2$$

4.11.4 Obtenção da solução ótima

Do Cálculo Elementar sabe-se que a aplicação da condição de otimalidade (restrições atendidas pelos pontos de máximo e mínimo de uma função diferenciável) permite obter a solução ótima do problema de otimização $\min J(\mathbf{w})$, na forma:

1. diferencie a função em relação aos parâmetros ajustáveis;
2. iguale estas derivadas parciais a zero;
3. resolva o sistema de equações resultante.

No caso em questão, os parâmetros livres são os coeficientes da combinação linear, dados na forma do vetor de $\mathbf{w} = [w_1 \ \dots \ w_j \ \dots \ w_m]^T$ pesos.

O sistema de equações resultante é dado na forma:

$$\frac{\partial J}{\partial w_j} = -2 \sum_{i=1}^N (s_i - f(\mathbf{x}_i)) \frac{\partial f}{\partial w_j} = -2 \sum_{i=1}^N (s_i - f(\mathbf{x}_i)) h_j(\mathbf{x}_i) = 0, \quad j=1, \dots, m.$$

Separando-se os termos que envolvem a incógnita $f(\cdot)$, resulta:

$$\sum_{i=1}^N f(\mathbf{x}_i) h_j(\mathbf{x}_i) = \sum_{i=1}^N \left[\sum_{r=1}^m w_r h_r(\mathbf{x}_i) \right] h_j(\mathbf{x}_i) = \sum_{i=1}^N s_i h_j(\mathbf{x}_i), \quad j=1, \dots, m.$$

Portanto, existem m equações para obter as m incógnitas $\{w_r, r=1, \dots, m\}$. Exceto sob condições “patológicas”, este sistema de equações vai apresentar uma solução única. Para encontrar esta solução única do sistema de equações lineares, é interessante recorrer à notação vetorial, fornecida pela álgebra linear, para obter:

$$\mathbf{h}_j^T \mathbf{f} = \mathbf{h}_j^T \mathbf{s}, \quad j=1, \dots, m,$$

onde

$$\mathbf{h}_j = \begin{bmatrix} h_j(\mathbf{x}_1) \\ \vdots \\ h_j(\mathbf{x}_N) \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \sum_{r=1}^m w_r h_r(\mathbf{x}_1) \\ \vdots \\ \sum_{r=1}^m w_r h_r(\mathbf{x}_N) \end{bmatrix} \quad \text{e} \quad \mathbf{s} = \begin{bmatrix} s_1 \\ \vdots \\ s_N \end{bmatrix}.$$

Como existem m equações, resulta:

$$\begin{bmatrix} \mathbf{h}_1^T \mathbf{f} \\ \vdots \\ \mathbf{h}_m^T \mathbf{f} \end{bmatrix} = \begin{bmatrix} \mathbf{h}_1^T \mathbf{s} \\ \vdots \\ \mathbf{h}_m^T \mathbf{s} \end{bmatrix}$$

Definindo a matriz $\mathbf{H}$, com sua j -ésima coluna dada por $\mathbf{h}_j$, temos:

$$\mathbf{H} = [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \dots \quad \mathbf{h}_m] = \begin{bmatrix} h_1(\mathbf{x}_1) & h_2(\mathbf{x}_1) & \dots & h_m(\mathbf{x}_1) \\ h_1(\mathbf{x}_2) & h_2(\mathbf{x}_2) & \dots & h_m(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_N) & h_2(\mathbf{x}_N) & \dots & h_m(\mathbf{x}_N) \end{bmatrix}$$

sendo possível reescrever o sistema de equações lineares como segue:

$$\mathbf{H}^T \mathbf{f} = \mathbf{H}^T \mathbf{s}$$

O i -ésimo componente do vetor $\mathbf{f}$ pode ser apresentado na forma:

$$f_i = f(\mathbf{x}_i) = \sum_{r=1}^m w_r h_r(\mathbf{x}_i) = [h_1(\mathbf{x}_i) \quad h_2(\mathbf{x}_i) \quad \dots \quad h_m(\mathbf{x}_i)] \mathbf{w}$$

permitindo expressar $\mathbf{f}$ em função da matriz $\mathbf{H}$, de modo que:

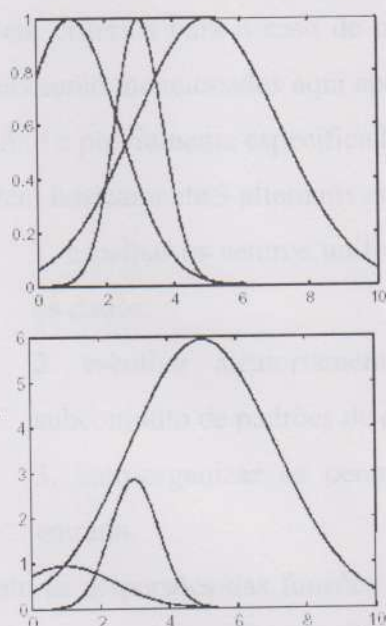
$$\mathbf{f} = \mathbf{H} \mathbf{w}$$

Substituindo no sistema de equações lineares, resulta a solução ótima para o vetor de coeficientes da combinação linear (que correspondem aos pesos da camada de saída da rede neural de base radial):

$$\mathbf{H}^T \mathbf{H} \mathbf{w} = \mathbf{H}^T \mathbf{s} \Rightarrow \mathbf{w} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{s}$$

Esta equação de solução do problema dos quadrados mínimos é conhecida como equação normal. Para que exista a inversa de $\mathbf{H}^T \mathbf{H}$, basta que a matriz $\mathbf{H}$ tenha posto completo, já que $m \leq N$.

4.11.5 Aproximação usando RNA RBF



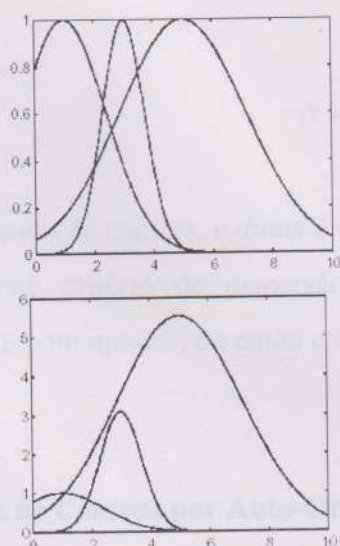
Caso 1: $m = N$

Pontos amostrados: (1,2); (3,7); (5,6)

$$\mathbf{c} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}; \quad \mathbf{r} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}; \quad \mathbf{w} = \begin{bmatrix} 0.945 \\ 2.850 \\ 5.930 \end{bmatrix}$$

Figura 34: aproximação usando RNA RBF caso 1.

Fonte: Nascimento; Yoneyama (2004).



Caso 2: $m < N$

Pontos amostrados: (1,2); (3,7); (5,6); (8,1)

$$\mathbf{c} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}; \quad \mathbf{r} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}; \quad \mathbf{w} = \begin{bmatrix} 1.012 \\ 3.084 \\ 5.538 \end{bmatrix}$$

Figura 35: aproximação usando RNA RBF caso 2.

Fonte: Nascimento; Yoneyama (2004).

4.11.6 Técnicas para Determinação dos Centros e Dispersões

No caso de algoritmos que se ocupam apenas com o ajuste dos pesos da camada de saída de uma rede RBF (modelos lineares nos parâmetros), é necessário estabelecer algum critério para fixação dos centros.

Existem critérios para o caso de número variável de centros (redes construtivas, por exemplo), mas serão mencionados aqui apenas aqueles geralmente empregados para o caso de um número fixo e previamente especificado de centros.

Existem basicamente 3 alternativas:

1. espalhar os centros uniformemente ao longo da região em que se encontram os dados;
2. escolher aleatoriamente, ou segundo algum critério específico, um subconjunto de padrões de entrada como centros;
3. auto-organizar os centros, de acordo com a distribuição dos dados de entrada.

Quanto às dispersões das funções de base radial, embora elas possam ser distintas (e até ajustáveis) para cada centro, usualmente se adota uma única dispersão para todos os centros, na forma (HAYKIN, 1999):

$$\rho = \frac{d_{\max}}{\sqrt{2m}}$$

onde m é o número de centros, e $d_{\max}$ é a distância máxima entre os centros.

Com este critério de dispersão, evita-se que as funções de base radial sejam excessivamente pontiagudas, ou então com uma base demasiadamente extensa.

4.11.7 Seleção de Centros por Auto-Organização

Para auto-organizar os centros, é suficiente aplicar algum algoritmo capaz de refletir a distribuição dos dados de entrada.

O algoritmo a ser apresentado a seguir, é um algoritmo de clusterização denominado *k*-means (BILLINGS; ZHENG, 1995). Este algoritmo se assemelha ao adotado para as redes de Kohonen, mas não leva em conta noções de vizinhança entre os centros.

Seja $\{c_j(t)\}_{j=1}^m$ os centros das funções de base radial na iteração t . O algoritmo *kmeans* pode ser descrito da seguinte forma:

1. *Inicialização*: Escolha valores aleatórios distintos para os centros $c_j(t)$.
2. *Amostragem*: Tome aleatoriamente um vetor x_i do conjunto de padrões de entrada;
3. *Matching*: Determine o índice k , do centro que mais se aproxima deste padrão, na forma: $k(x_i) = \arg \min_j \|x_i(t) - c_j(t)\|$, $j = 1, \dots, m$.
4. *Ajuste*: Ajuste os centros usando a seguinte regra:

$$c_j(t+1) = \begin{cases} c_j(t) + \gamma[x_i(t) - c_j(t)], & k = k(x_i) \\ c_j(t), & \text{alhores} \end{cases}$$

onde $\gamma \in (0,1)$ é a taxa de ajuste.

5. *Ciclo*: Repita os Passos 2 a 5 para todos os N padrões de entrada e até que os centros não apresentem deslocamento significativo após cada apresentação completa dos N padrões.

5.1 TIPO DE PESQUISA

5.1.1 Quanto aos Objetivos

A pesquisa foi classificada quanto aos objetivos como *exploratória*, que é o primeiro passo a ser feito em todo trabalho científico, tendo como finalidade proporcionar maiores informações sobre o assunto; facilitar a delimitação do tema, definir os objetivos ou formular as hipóteses de uma pesquisa ou descobrir quais tipos de enfoque para o trabalho que se tem em mente. Portanto, a pesquisa exploratória, na maioria dos casos, envolve um trabalho preliminar ou preliminar (ANDRADE, 1996).

5 METODOLOGIA

5.1.2 Quanto aos Procedimentos

Para a realização deste trabalho realizou-se uma extensa pesquisa exploratória com intuito de obter conhecimento necessário para o desenvolvimento do algoritmo da RNA RBF, capaz de aprender funcionalidades dos operadores lógicos (*XOR*, *AND* e *OR*) para a análise do comportamento da rede em relação a variação arquitetural nos referidos operadores lógicos, que utiliza a linguagem de programação *Java* com a tecnologia *Swing*.

O presente trabalho trata-se de uma pesquisa aplicada, onde serão seguidos, os seguintes passos:

- Realizar um estudo sobre redes neurais artificiais, analisando a aplicabilidade de técnicas no processo de reconhecimento de padrões;
- Realizar um estudo sobre os operadores lógicos que serão utilizados na implementação da RNA RBF;
- Analisar dados que serão utilizados na alimentação da RNA;
- Implementar a rede neural;
- Validar o modelo implementado;
- Treinar a RNA para cada operador lógico (*XOR*, *AND* e *OR*);
- Realizar testes e análise arquitetural e comportamental RNA.

5.1 TIPO DE PESQUISA

5.1.1 Quanto aos Objetivos

A pesquisa foi classificada quanto aos objetivos como exploratória, que é o primeiro passo a ser feito em todo trabalho científico, tendo como finalidades proporcionar maiores informações sobre o assunto; facilitar a delimitação do tema; definir os objetivos ou formular as hipóteses de uma pesquisa ou descobrir novo tipo de enfoque para o trabalho que se tem em mente. Portanto, a pesquisa exploratória, na maioria dos casos, constitui um trabalho preliminar ou preparatório (ANDRADE, 1998).

6 RESULTADOS

5.1.2 Quanto aos Procedimentos

O trabalho trata-se de uma pesquisa bibliográfica com intuito de obter dados suficientes para a realização do projeto. A pesquisa bibliográfica abrange toda a bibliografia já tornada pública, referente ao assunto de estudo, como publicações avulsas, boletins, jornais, revistas, livros, pesquisas, monografias, teses, etc., até meios de comunicação orais (ANDRADE, 1998).

5.1.3 Quanto ao Objeto

A pesquisa foi classificada como de laboratório, onde serão utilizadas as ferramentas de desenvolvimento, para a construção de uma RNA RBF. A pesquisa de laboratório descreve e analisa o que será ou ocorrerá em situações controladas. Exige instrumental específico, preciso, e ambientes adequados (ANDRADE, 1998).

6 RESULTADOS

Como resultado da pesquisa foi desenvolvido um protótipo de uma Rede Neural Artificial RBF, utilizando a plataforma *Java Standard Edition* (JSE) e a API *Java Swing*. Como ferramentas de desenvolvimento utilizou-se a IDE *netbeans* 6.0.1, a máquina virtual *Java* 6.0, da *Sun Microsystem*. Devido à necessidade de se utilizar desenhos para a representação de gráficos, foi utilizado o *Canvas*, para desenhar os gráficos de convergência.

A aplicação foi desenvolvida em uma arquitetura na qual está dividida em três etapas:

- a) Configuração - na qual defini-se a taxa de aprendizagem, o erro médio aceitável, erro de correção, a quantidade de neurônios na camada escondida e o operador lógico que será tomado como base para o aprendizado;
- b) Treinamento - onde é realizado o aprendizado da rede, apresentando o gráfico do erro médio e o número de iterações feitas pelo algoritmo;
- c) Teste - utilizou-se a rede para a operação lógica definida, anteriormente, visando o aprendizado.

O diagrama da figura 36 ilustra como as classes da aplicação se relacionam.

O código está distribuído em 5 classes (*RBF.java*, *NeuronioEscondido.java*, *RBFGui.java*, *Treinamento.java*, *Usar.java*) que estão no pacote *IA_RBF*, onde estão as classes responsáveis pelo funcionamento da RNA RBF.



Figura 36: Diagrama de classes da aplicação

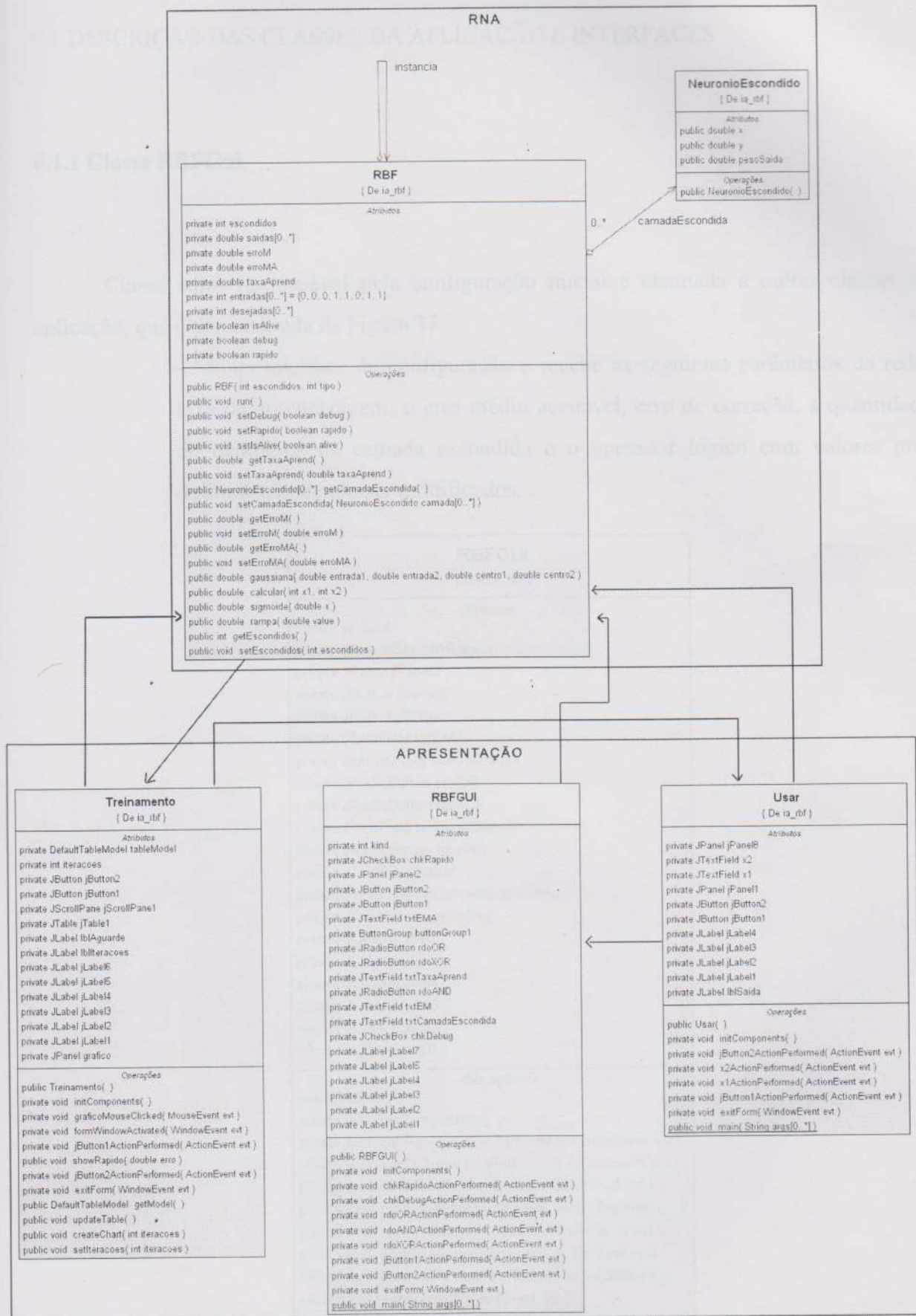


Figura 36: diagrama da aplicação.
Fonte: Dados da Pesquisa (2008)

6.1 DESCRIÇÃO DAS CLASSES DA APLICAÇÃO E INTERFACES

6.1.1 Classe RBF

6.1.1 Classe RBFGui

Classe *main* responsável pela configuração inicial e chamada a outras classes da aplicação, que é demonstrada na Figura 37:

- Monta interface de configuração e recebe os seguintes parâmetros da rede: taxa de aprendizagem, o erro médio aceitável, erro de correção, a quantidade de neurônios na camada escondida e o operador lógico com valores pré-definidos, podendo ser modificados.

RBFGUI { De ia_rbf }	
Atributos	
private int kind	
private JCheckBox chkRapido	
private JPanel jPanel2	
private JButton jButton2	
private JButton jButton1	
private JTextField txtEMA	
private ButtonGroup buttonGroup1	
private JRadioButton rdoOR	
private JRadioButton rdoXOR	
private JTextField txtTaxaAprend	
private JRadioButton rdoAND	
private JTextField txtEM	
private JTextField txtCamadaEscondida	
private JCheckBox chkDebug	
private JLabel jLabel7	
private JLabel jLabel5	
private JLabel jLabel4	
private JLabel jLabel3	
private JLabel jLabel2	
private JLabel jLabel1	
Operações	
public RBFGUI()	
private void initComponents()	
private void chkRapidoActionPerformed(ActionEvent evt)	
private void chkDebugActionPerformed(ActionEvent evt)	
private void rdoORActionPerformed(ActionEvent evt)	
private void rdoANDActionPerformed(ActionEvent evt)	
private void rdoXORActionPerformed(ActionEvent evt)	
private void jButton1ActionPerformed(ActionEvent evt)	
private void jButton2ActionPerformed(ActionEvent evt)	
private void exitForm(WindowEvent evt)	
public void main(String args[0..*])	

Figura 37: classe RBFGui.

Fonte: Dados da Pesquisa (2008).

6.1.2 Classe RBF

Classe onde está contida parte principal da lógica do protótipo da RNA RBF, como podemos observar na Figura 38:

- Recebe as informações da interface de configuração e inicializa os seguintes parâmetros da RNA RBF: taxa de aprendizagem, o erro médio aceitável, erro de correção, a quantidade de neurônios na camada escondida, operador lógico, entradas, desejadas, debug, rapido.

RBF { De ia_rbf }	
<i>Atributos</i>	
<pre>private int escondidos private double saidas[0..*] private double erroM private double erroMA private double taxaAprend private int entradas[0..*] = {0, 0, 0, 1, 1, 0, 1, 1} private int desejadas[0..*] private boolean isAlive private boolean debug private boolean rapido</pre>	
<i>Operações</i>	
<pre>public RBF(int escondidos, int tipo) public void run() public void setDebug(boolean debug) public void setRapido(boolean rapido) public void setIsAlive(boolean alive) public double getTaxaAprend() public void setTaxaAprend(double taxaAprend) public NeuronioEscondido[0..*] getCamadaEscondida() public void setCamadaEscondida(NeuronioEscondido camada[0..*]) public double getErroM() public void setErroM(double erroM) public double getErroMA() public void setErroMA(double erroMA) public double gaussiana(double entrada1, double entrada2, double centro1, double centro2) public double calcular(int x1, int x2) public double sigmoide(double x) public double rampa(double value) public int getEscondidos() public void setEscondidos(int escondidos)</pre>	

Figura 38: classe RBF.

Fonte: Dados da Pesquisa (2008).

Nesta Classe RBF existem alguns métodos de extrema importância para a RNA RBF como pode-se observar no Algoritmos 06 e Algoritmo 07.

```
public double gaussiana ( double entrada1, double entrada2, double centro1, double centro2 ) {

    double distancia1, distancia2;
    distancia1 = Math.abs( entrada1 - centro1 );
    distancia1*=distancia1;
    double variancia = 1;

    distancia2 = Math.abs( entrada2 - centro2 );
    distancia2*= distancia2;

    double norma = Math.sqrt( distancia1 + distancia2 );
    return Math.exp( - ( norma*norma / 2*variancia*variancia) );

}
```

Algoritmo 06: função gaussiana.

Fonte: Dados da Pesquisa (2008).

```
public double sigmoide ( double x ) { // Metodo sigmoide

    return 1 / ( 1 + Math.exp( -x ) );

}
```

Algoritmo 07: função sigmoid.

Fonte: Dados da Pesquisa (2008).

6.1.3 Classe Neurônio Escondido

Classe onde está contida a menor parte de uma rede neural o neurônio, como podemos observar na Figura 39 e no Algoritmo 08:

- Uma vez instanciada, esta classe inicializa os parâmetros x, y e pesoSaida do novo neurônio com valores randômicos através de um método construtor.

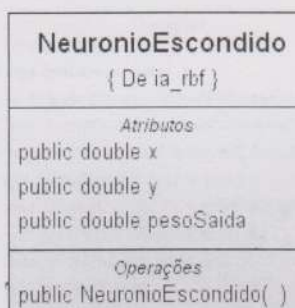


Figura 39: classe NeuronioEscondido.

Fonte: Dados da Pesquisa (2008).

```

class NeuronioEscondido { //Classe do Neuronio escondido
    public double x; //Valor de x
    public double y; //Valor de y
    public double pesoSaida; //Peso de saida

    public NeuronioEscondido() { //método construtor da classe NeuronioEscondido
        x = Math.random(); //Atribui a variável x um valor randomico inicial
        y = Math.random(); //Atribui a variável y um valor randomico inicial
        pesoSaida = Math.random(); //Atribui a variável pesoSaida um valor randomico inicial
    }
}

```

Algoritmo 08: NeuronioEscondido.

Fonte: Dados da Pesquisa (2008).

6.1.4 Classe Treinamento

Esta Classe monta a interface de treinamento onde podemos observar o número de iterações em tempo real e o gráfico da curva de aprendizado após a convergência como nota-se na Figura 40 e no Algoritmo 09.

Treinamento { De ia_rbf }
<i>Atributos</i> <pre> private DefaultTableModel tableModel private int iteracoes private JButton jButton2 private JButton jButton1 private JScrollPane jScrollPane1 private JTable jTable1 private JLabel lblAguarde private JLabel lblIteracoes private JLabel jLabel6 private JLabel jLabel5 private JLabel jLabel4 private JLabel jLabel3 private JLabel jLabel2 private JLabel jLabel1 private JPanel grafico </pre>
<i>Operações</i> <pre> public Treinamento() private void initComponents() private void graficoMouseClicked(MouseEvent evt) private void formWindowActivated(WindowEvent evt) private void jButton1ActionPerformed(ActionEvent evt) public void showRapido(double erro) private void jButton2ActionPerformed(ActionEvent evt) private void exitForm(WindowEvent evt) public DefaultTableModel getModel() public void updateTable() public void createChart(int iteracoes) public void setIteracoes(int iteracoes) </pre>

Figura 40: classe Treinamento.

Fonte: Dados da Pesquisa (2008).

```

public void createChart( int iteracoes ) {

    lblAguarde.setVisible(false);

    this.iteracoes = iteracoes;

    int height = grafico.getHeight();
    int salto = 1, x = 0;
    salto = iteracoes / height - 1;
    if ( salto == 0 )
        salto = 1;

    int x_ant = 0, ypos_ant = 0;

    for ( int a = 0; a < tableModel.getRowCount(); a++ ) {
        double value = new Double( (String) tableModel.getValueAt(a*salto,0) ).doubleValue();
        int ypos = (int) ( height - value*height );
        grafico.getGraphics().drawLine( x_ant, ypos_ant, x + 1, ypos );
        x_ant = x;
        ypos_ant = ypos;
        x+=1;
    }
}

```

Algoritmo 09: método createChart (método para o desenho do gráfico).

Fonte: Dados da Pesquisa (2008).

6.1.5 Classe Usar

Esta Classe monta a interface de testes, onde são informados como entrada valores binários 0 ou 1, a fim de verificar se a RNA RBF realmente aprendeu as funcionalidades dos operadores lógicos *XOR*, *OR* e *AND* como verifica-se na Figura 41 e no Algoritmo 10 e 11.

Usar { De ia_rbf }
Atributos
private JPanel jPanel8; private JTextField x2; private JTextField x1; private JPanel jPanel1; private JButton jButton2; private JButton jButton1; private JLabel jLabel4; private JLabel jLabel3; private JLabel jLabel2; private JLabel jLabel1; private JLabel lblSaida
Operações
public Usar() private void initComponents() private void jButton2ActionPerformed(ActionEvent evt) private void x2ActionPerformed(ActionEvent evt) private void x1ActionPerformed(ActionEvent evt) private void jButton1ActionPerformed(ActionEvent evt) private void exitForm(WindowEvent evt). public void main(String args[0..*])

Figura 41: classe Usar.

Fonte: Dados da Pesquisa (2008).

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // add your handling code here:
    double saida = RBF.instancia.calcular( Integer.parseInt( x1.getText() )
                                           , Integer.parseInt( x2.getText() ) );
    lblSaida.setText( String.valueOf( saida ) );
}
```

Algoritmo 10: método jButton1ActionPerformed (método de chamada do método calcular na classe RBF).
Fonte: Dados da Pesquisa (2008).

```
public double calcular ( int x1, int x2 ) {

    double saidasEscondidas[] = new double[escondidos];
    double saida = 0.0;

    for ( int a = 0; a < escondidos; a++ ) {
        NeuronioEscondido neuronio = camadaEscondida[a];
        //saidasEscondidas[a] = gaussiana( x1, neuronio.x, x2, neuronio.y );
        saidasEscondidas[a] = sigmoide( (double) x1*neuronio.x
                                       + (double) x2*neuronio.y );
        saida += saidasEscondidas[a] * neuronio.pesoSaida;
    }

    return sigmoide(saida);
}
```

Algoritmo 11: método calcular(método calcular na classe RBF).
Fonte: Dados da Pesquisa (2008).

6.2 INTERFACES DA APLICAÇÃO RNA RBF

Neste protótipo trabalha-se basicamente com três interfaces: configuração, treinamento e teste como podemos observar a seguir nas Figuras 42, 43 e 44.



Figura 41: interface de Configuração.
Fonte: Dados da Pesquisa (2008).

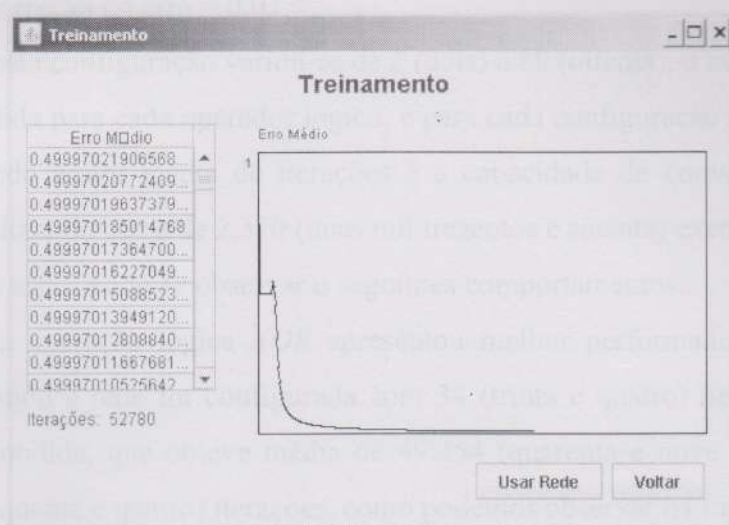


Figura 42: interface de Treinamento.
Fonte: Dados da Pesquisa (2008).

Usando a Rede

Entradas

Entrada 1: 1

Entrada 2: 1

Calcular

Voltar

Saída

0.003737410379726541

Figura 43: interface de Utilização da RNA.
Fonte: Dados da Pesquisa (2008).

6.3 TESTES E ANALISE COMPORTAMENTAL DA RNA RBF

Realizou-se inúmeros testes com a RNA RBF por meio de parametrizações e configurações, com o intuito de observar o comportamento da rede em relação à sua camada escondida. Desenvolveu-se a seguinte estratégia de teste para os operadores lógicos *XOR*, *OR* e *AND*, para a análise comportamental da RNA RBF que teve a seguinte configuração:

- Taxa de aprendizagem = 0.3;
- Erro médio aceitável = 0.01;

- Correção do erro = 0.01.

Seguindo esta configuração variou-se de 2 (dois) a 80 (oitenta), o número de neurônios da camada escondida para cada operador lógico, e para cada configuração foram efetuados 10 (dez) testes obtendo assim média de iterações e a capacidade de convergência com cada configuração, totalizando cerca de 2.370 (duas mil trezentos e setenta) execuções.

Após estes testes pode-se observar o seguintes comportamentos:

a) a operação lógica *XOR* apresentou melhor performance na convergência quando a rede foi configurada com 34 (trinta e quatro) neurônios na camada escondida, que obteve média de 49.454 (quarenta e nove mil quatrocentos e cinquenta e quatro) iterações, como podemos observar na Figura 44;

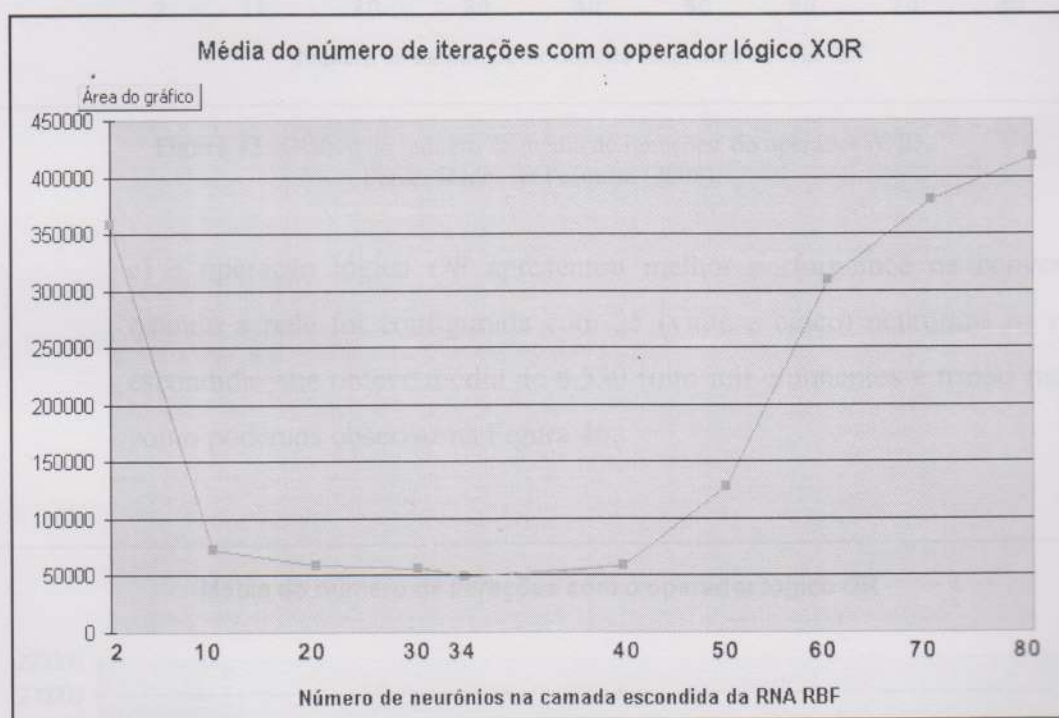


Figura 44: Gráfico do número da média de iterações do operador XOR.

Fonte: Dados da Pesquisa (2008).

b) a operação lógica *AND* apresentou melhor performance na convergência quando a rede foi configurada com 11 (onze) neurônios na camada escondida, que obteve média de 16.310 (dezesesseis mil trezentos e dez) iterações, como podemos observar na Figura 45;

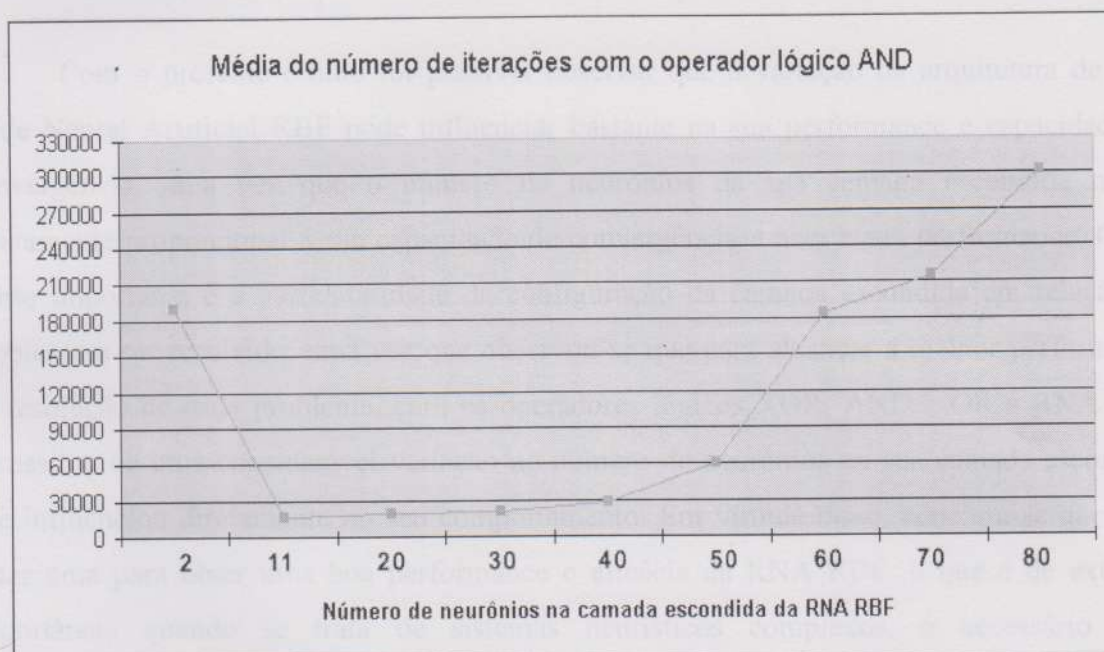


Figura 45: Gráfico do número da média de iterações do operador AND.

Fonte: Dados da Pesquisa (2008).

c) a operação lógica *OR* apresentou melhor performance na convergência quando a rede foi configurada com 25 (vinte e cinco) neurônios na camada escondida, que obteve média de 8.530 (oito mil quinhentos e trinta) iterações, como podemos observar na Figura 46;

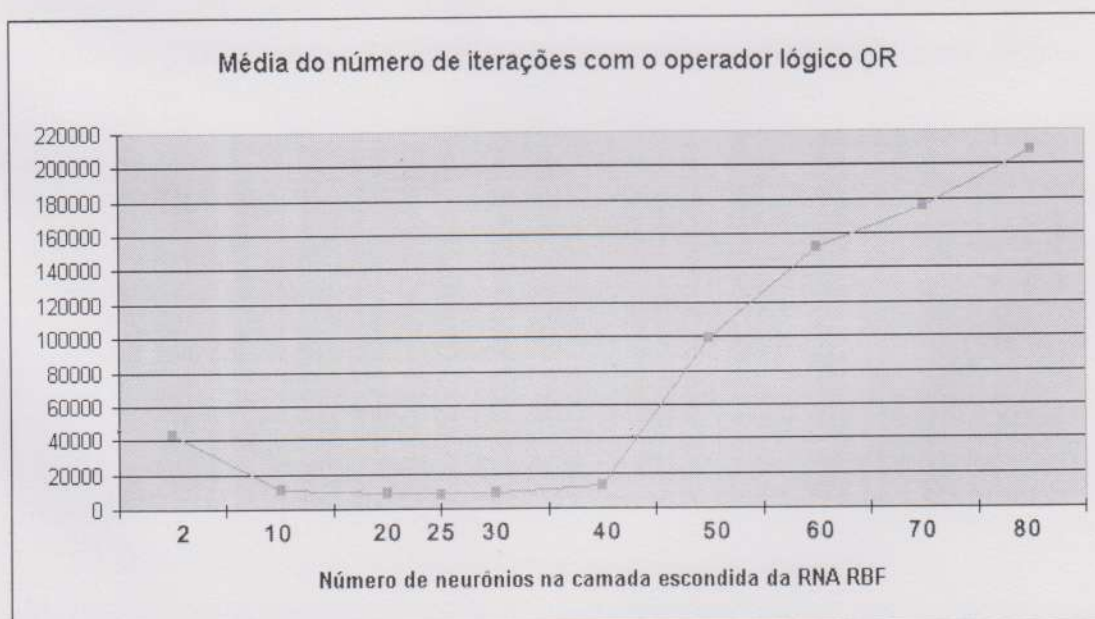


Figura 45: Gráfico do número da média de iterações do operador OR.

Fonte: Dados da Pesquisa (2008).

7 CONCLUSÃO

Com o presente estudo foi possível observar que a variação da arquitetura de uma Rede Neural Artificial RBF pode influenciar bastante na sua performance e capacidade de convergência, uma vez que o número de neurônios da sua camada escondida não é diretamente proporcional a sua capacidade de convergência e nem a sua performance. Outro ponto importante é a particularidade da configuração da camada escondida em relação ao problema a ser resolvido, uma vez que observou-se que para alcançar a melhor performance na resolução de cada problema, com os operadores lógicos XOR, AND e OR a RNA RBF necessitou de uma considerável variação no número de neurônios na sua camada escondida que influenciou diretamente no seu comportamento. Em virtude disso, concluiu-se que para obter uma boa performance e eficácia da RNA RBF, o que é de extrema importância quando se trata de sistemas heurísticos complexos, é necessário uma configuração precisa em relação à quantidade de neurônios na sua camada escondida, pois uma configuração mal estruturada acarretará no aumento do custo computacional e, em determinadas vezes, inviabiliza o processo de convergência.

RUZICKA, S. A., *Neural networks: a comprehensive foundation*, New York: Macmillan, 1990.

HECCHER, NIELSEN, R., *Neurocomputers*, Addison-Wesley, p. 432, Massachusetts, 1990.

HOLLAND, T.R.; MORASKI, M., *ELSA Neural networks*, Comput. Chem. Engng., v. 16, n. 4, p. 393-411, 1992.

KOHOPSEN, T., *Correlation matrix memories*, IEEE C-31, p. 353-359, 1971.

LEE, S.; CHILKOTI, M.R., *A gaussian potential function network with biologically self-organizing learning*, Neural Networks, v. 4, p. 207-234, 1991.

HEPPELMANN, R.P., *Comparisons between neural net and conventional classifiers*, IEEE International Conference on Neural Networks, San Diego, 1997.

LIORRE, G.F.; STURGEFIELD, W.A., *Artificial intelligence: Schemes and strategies for complex problem solving*, Benjamin Cummings, Redwood City, California, second edition, 1992.

REFERÊNCIAS

- ACKLEY, D. H.; Hinton, G. E.; SEJNOWSKI, T. J.; A learning algorithm for boltzmann machines, **Cognitive Science**, 9, 147-169, 1985.
- BILLINGS, S.A.; ZHENG, G.L., Radial basis function networks configuration using genetic algorithms, **Neural Networks**, v. 8, n. 6, p. 877-890, 1995.
- CHEN, C.-L.; CHEN, W.-C. ; CHANG, F.-Y. Hybrid learning algorithm for Gaussian potential function networks, **IEEEProceedings D**, 442-448, 1991.
- FAUSETT, L., Fundamentals of neural networks: architectures, algorithms, and applications, **Prentice Hall**, 1994.
- GALHARDO, Raphaela F., Detecção e isolamento de falhas em sistemas dinâmicos baseados em redes neurais, (Dissertação de Mestrado) Pós-Graduação em Engenharia Elétrica da UFRN, 2007.
- HAYKIN, S. S., Neural networks: a comprehensive foundation, New York: **Macmillan**, 1999.
- HECHT-NIELSEN, R., Neurocomputers, **Addison-Wesley**, p. 432, Massachusetts, 1990.
- HOLCOMB, T.R.; MORARI, M., PLS/Neural networks, **Comput. Chem. Engng**, v. 16, n. 4, p. 393-411, 1992.
- KOHONEN, T., Correlation matrix memories, **IEEE C-21**, p. 353-359, 1972.
- LEE, S.; RHEE, M.K., A gaussian potencial function network with hierarchically self-organizing learning", **Neural Networks**, v. 4, p. 207-224, 1991.
- LIPPMANN, R.P., Comparisons between neural net and conventional classifiers. **IEEE International Conference on Neural Networks**, San Diego, 1997.
- LUGER, G.F; STUBBLEFIELD, W.A., Artificial intelligence: Structures and strategies for complex problem solving. **Benjamim/Cummings**, Redwood City, California, second edition. 1993.

MENDES, Filho, Redes neurais artificiais 1998. Disponível em: <<http://www.icmsc.sc.usp.br/~prico/neural1.html#topicos>> Acesso em: 12 de jun. 2008.

MOCLOONE, S., BROWN, M.D., IRWIN, L., A hybrid linear/nonlinear training algorithm for feedforward neural networks", **IEEE Transactions on Neural Networks**, v. 9, n. 4, July 1998.

MUSAVI, M.T.; AHMED, W.; CHAN, K.H.; FARIS, K.B; HUMMELS, D.M., On the training of radial basis function classifiers, **Neural Networks**, v. 5, p. 595-603, 1992.

NASCIMENTO, Cairo L. J.; YONEYAHMA, Takasshi, Inteligência artificial em controle e automação, **Editora Edgard Blucher**, v.1, 2004.

NEVES, J. C., Estimating bankruptcy using neural networks trained with hidden layer learning vector quantization. **Working Paper**, 2004.

RABUSKE, Renato A., Inteligência artificial. Florianópolis, **Editora da UFSC**, 1995.

ROSENBLATT, F., Principles of neurodynamics, **Spartan Books**, 1962.

RUSSEL, S. J.; NORVIG, P., Inteligência artificial., v.2, Rio de Janeiro: **Elsevier**, 2004.

SCHALKOFF, R. J., Artificial intelligence: an engineering approach. **McGraw-Hill**, New York. 1990.

STARKE, J. F., Comparativo entre modelos de redes neurais aplicadas ao reconhecimento de fala. **FRUB**, Blumenau, 1995.

TAFNER, M. A.; XEREZ, M.; RODRIGUES FILHO., I. W., Redes neurais artificiais - princípios de Neurocomputação. **EKO e FURB**, Blumenau. 1995.

TODESCO, José L., Reconhecimento de padrões usando rede neuronal artificial com uma função de base radial: uma aplicação na classificação de cromossomos humanos., **UFSC** Florianópolis, 1995.

WINSTON, N., Problemas e hipóteses., **Hoplt, Reineher and Winston**. New York, 1984.