

Trabalho de Conclusão de Curso

Engenharia da Computação

iMaRag: Um algoritmo baseado em Sistemas Imunológicos Artificiais para treinamento de redes MLP

Gilliard Alan de Melo Lopes

Orientador: Prof. Dr. Mêuser Jorge Silva Valença



UNIVERSIDADE
DE PERNAMBUCO

Gilliard Alan de Melo Lopes

**iMaRag: Um algoritmo baseado em
Sistemas Imunológicos Artificiais para
treinamento de redes MLP**

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, junho de 2010.

À todos que me amam e desejam meu bem.

Agradecimentos

Antes de qualquer coisa, agradeço a Deus por me dar saúde e paz.

Agradeço aos meus pais: Marcos Eugênio Lopes Fonseca e Maria Lúcia Marques de Melo Lopes e também as minhas irmãs: Gilianne Fernanda e Gigliola Maria e ao meu cunhado: Daniel Valpassos, todos vocês fazem parte de minha história pelos conselhos, valores morais e cuidados que sempre me proporcionaram.

Agradeço a minha namorada: Nathalia Maria Temudo, pelo apoio e suporte, sobretudo no desenvolvimento deste trabalho e fora dele, pelo companheirismo, carinho e atenção.

Agradeço a todos meus amigos, em especial aos mais próximos que sempre pude contar: Carlos Bruno, Igor Feitosa, João Fausto, Rafael Galvão, Diego Siqueira, Arthur Minduca, Caio Cesar, Saulo Medeiros e Lorena Tablada.

Ao meu orientador, além de agradecimentos só tenho elogios a fazer por toda sua atenção, prestação e tratamento comigo.

Agradeço a todos os professores que tive durante a graduação, sem eles eu não estaria me sentindo tão preparado para resolver os problemas que com certeza encontrarei na minha carreira profissional e acadêmica.

“Se, a princípio, a idéia não é absurda, então não há esperança para ela”.

Albert Einstein

Resumo

O objetivo deste trabalho é implementar e explorar a utilização de um algoritmo baseado em Sistemas Imunológicos Artificiais (SIA) para o treinamento de uma Rede Neural Artificial (RNA) do tipo Multi-Layer Perceptron (MLP). Os Sistemas Imunológicos Artificiais são algoritmos bio-inspirados, ou seja, algoritmos inspirados em fenômenos da natureza. Existem diversos modelos de algoritmos baseados em SIAs e cada um procura resolver um tipo de problema do mundo real. Recentemente, um desses modelos, o de seleção clonal, vem sendo utilizado em diversos problemas de busca e otimização. No trabalho, foi construído um algoritmo com base no modelo de seleção clonal dos SIAs. O nome dado ao algoritmo é iMaRag, que é uma abreviação de *Immune Artificial Algorithm*. Também foi implementado e utilizado o algoritmo *backpropagation*, que é o algoritmo tradicional para o treinamento de uma RNA. Dessa maneira, é feita uma comparação entre os resultados dos dois algoritmos, levando em consideração a precisão na taxa de acerto da RNA, para o caso das bases de classificação, e o Erro Percentual Médio Absoluto (EMPA) para as bases de regressão.

Palavras-chave: Redes Neurais Artificiais, Algoritmos de treinamento para redes MLP e Sistemas Imunológicos Artificiais.

Abstract

The main goal of this project is to develop and explore the utilization of an algorithm that is based on Artificial Immune Systems for training Multi-Layer Perceptron Artificial Neural Networks. The Artificial Immune Systems are bio-inspired algorithms, in other words, algorithms inspired by natural phenomena. There are several models of algorithms based on SIAs and each one tries to solve some kind of real world problem. Recently, one of these models, clonal selection, has been used in many problems, more specifically in search and optimization problems. At this work, we developed an algorithm based on clonal selection model from Artificial Immune Systems. The name of this algorithm is *iMaRag*, which is an abbreviation of *Immune Artificial Algorithm*. The backpropagation, which is the traditional algorithm for training an artificial neural network, was also implemented and used in this work. Thus, a comparison is made between the results of the two algorithms, considering the precision in the accuracy rate of the artificial neural network, in the case of the classification datasets, and Mean Absolute Percentage Error in the case of the regression datasets.

Key-words: Artificial Neural Networks, Training algorithms to MLP networks, Artificial Imunne Systems,

Sumário

Conteúdo

Agradecimentos	iv
Resumo	vi
Abstract	vii
Sumário	viii
Índice de Figuras	xi
Índice de Tabelas	xii
Tabela de Símbolos e Siglas	xiii
1. Introdução	14
1.1 Definição do problema.....	14
1.2 Objetivos	15
1.3 Estrutura da monografia	15
2. Redes Neurais Artificiais (RNAs)	17
2.1 Modelo biológico do neurônio.....	17
2.2 Modelo computacional do neurônio.....	20
2.3 Regra de Aprendizado.....	23
2.3.1 Aprendizado com um Supervisor	23
2.3.2 Aprendizado sem um Supervisor	24
2.4 Multi-Layer Perceptron (MLP).....	25
2.4.1 Perceptron.....	25
2.4.2 Arquitetura.....	26
2.4.3 Características	27
2.5 Algoritmo Backpropagation	28
2.5.1 Fase forward	28

2.5.2	Fase backward	29
2.6	Validação Cruzada (VC).....	30
3.	Sistemas Imunológicos	32
3.1	Sistema Imunológico Biológico.....	32
3.1.1	Sistema imune adaptativo	34
3.2	Sistemas Imunológicos Artificiais (SIAs)	36
3.3	Seleção Clonal	38
3.3.1	Critério de dominância.....	39
4.	Metodologia	41
4.1	Bases de dados.....	41
4.1.1	Base Íris	41
4.1.2	Base de Vinhos	42
4.1.3	Base de Curuá-Una.....	42
4.2	Tratamento dos dados.....	42
4.3	Processamento da rede MLP com o <i>backpropagation</i>	43
4.4	O algoritmo proposto: <i>iMaRag</i>	46
4.4.1	Representação/Mapeamento do problema	46
4.4.2	Inicialização do repertório.....	47
4.4.3	Sensibilidade à semente de geração dos pesos aleatórios.....	48
4.4.4	Inicialização do treinamento da rede MLP.....	49
4.4.5	Atribuição de <i>fitness</i> e cálculo do <i>rank</i>	50
4.4.6	Clonagem	51
4.4.7	Maturação	52
4.4.8	Otimização dos parâmetros e ajuste de fórmulas do modelo original para o <i>iMaRag</i>	54
5.	Resultados	57

5.1	Problemas de classificação	57
5.1.1	Base Íris	57
5.1.2	Base de Vinhos	59
5.2	Problemas de Regressão	61
5.2.1	Base de Curuá-Una.....	61
6.	Conclusão	63
6.1	Objetivos Futuros	63
	Referências	65

Índice de Figuras

Figura 1.	Neurônio e seus componentes	19
Figura 2.	Sinapse entre dois neurônios.	19
Figura 3.	Neurônio de MCP com limiar explícito (VALENÇA, Mêuser, 2009).....	21
Figura 4.	Neurônio de MCP com limiar implícito (VALENÇA, Mêuser, 2009).....	21
Figura 5.	Esquema de aprendizado supervisionado.....	24
Figura 6.	Arquitetura do <i>perceptron</i>	26
Figura 7.	Exemplo de problema linearmente separável	26
Figura 8.	Arquitetura de uma rede MLP com duas camadas intermediárias	27
Figura 9.	Fluxo do <i>backpropagation</i> : propagação para frente dos sinais de entrada e retropropagação de sinais de erro (HAYKIN, Simon, 2001)	29
Figura 10.	Critério de parada do treinamento baseado na VC	31
Figura 11.	Arquitetura multicamadas do sistema imunológico (DE CASTRO, L. N., 2001)	33
Figura 12.	O sistema imune adaptativo (DE CASTRO, L. N., 2001)	35
Figura 13.	Exemplos de modelos dos SIAs	37
Figura 14.	Configuração da arquitetura da rede e do <i>backpropagation</i>	44
Figura 15.	Parâmetros de configuração (inicialmente) para o iMaRag	46
Figura 16.	Arquitetura de rede MLP e seus pesos sinápticos.....	47
Figura 17.	Número de clones gerados por anticorpo em função do Rank(Ab_i)	51
Figura 18.	Processo de maturação no ajuste dos pesos dos clones.....	53
Figura 19.	Curva gerada para o fator exponencial utilizado no cálculo de α_i	55
Figura 20.	Esquema do iMaRag	56

Índice de Tabelas

Tabela 1.	Alguns exemplos de funções de ativação	22
Tabela 2.	Modelos de SIA e suas resoluções práticas (DASGUPTA, D., 2006)	37
Tabela 3.	Exemplo de repertório de anticorpos gerado para um espaço de busca no intervalo [-30;30].....	48
Tabela 4.	Variação das sementes de geração de pesos aleatórios e resultados encontrados na base de Íris	48
Tabela 5.	Configuração da rede MLP nas simulações da base de íris e vinhos	57
Tabela 6.	Resultados obtidos pela rede MLP com o algoritmo de treinamento <i>backpropagation</i>	58
Tabela 7.	Resultados obtidos pela rede MLP com o algoritmo de treinamento iMaRag	59
Tabela 8.	Resultados obtidos pela rede MLP com o algoritmo de treinamento <i>backpropagation</i>	60
Tabela 9.	Resultados obtidos pela rede MLP com o algoritmo de treinamento iMaRag	60
Tabela 10.	Configuração da arquitetura da rede MLP para a base de Curuá-Una	61
Tabela 11.	Resultados obtidos pela rede MLP com o algoritmo de treinamento <i>backpropagation</i>	62
Tabela 12.	Resultados obtidos pela rede MLP com o algoritmo de treinamento iMaRag	62

Tabela de Símbolos e Siglas

(Dispostos por ordem de aparição no texto)

SIA – Sistema Imunológico Artificial

RNA – Rede Neural Artificial

MLP – Multi-Layer Perceptron

iMaRag – Immune Artificial Algorithm

CI – Computação Inteligente

MCP - McCulloch e Pitts

EMQ – Erro Médio Quadrático

VC – Validação Cruzada

EMQVC - Erro Médio Quadrático de Validação Cruzada

EMPA – Erro Médio Percentual Absoluto

1. Introdução

As RNAs são modelos bio-inspirados que constituem um ramo da Computação Inteligente (CI) e que vêm sendo amplamente utilizados e melhorados ao longo dos últimos anos. As RNAs se baseiam, de maneira geral, no funcionamento do cérebro humano e diferentemente da arquitetura de Von Neumann, utilizada atualmente nos computadores, o cérebro humano é uma estrutura que possui um processamento extremamente paralelo e que mesmo formado por unidades (neurônios) que trabalham individualmente de maneira lenta, é capaz de realizar o processamento de trilhões de operações concomitantemente. Além disso, as RNAs possuem características como capacidade de aprendizagem, habilidade para generalização, adaptação, entre outras, o que as torna capazes de resolver muitos problemas complexos do mundo real.

Os SIAs também são modelos bio-inspirados e que constituem uma ramificação da CI. Durante a última década, o interesse no desenvolvimento de modelos computacionais inspirados nos princípios imunológicos vem aumentando significativamente. Inclusive, já existem diversos modelos computacionais que simulam mecanismos e processos do sistema imunológico biológico para resolução de problemas como: detecção de falta, classificação, clusterização, otimização de funções, segurança de redes, etc (DASGUPTA, Dipankar, 2006). Ao contrário das RNAs, os SIAs constituem um campo da CI relativamente novo, porém bastante atrativo, com modelos, técnicas e aplicações de grande diversidade.

1.1 Definição do problema

A RNA MLP é uma técnica conhecida por ser um aproximador universal de funções. Uma etapa de fundamental importância quando da utilização desta técnica está no **treinamento da rede**. O treinamento da rede neural consiste no ajuste dos pesos de forma a dotá-la de capacidade de generalização. Nesse contexto, o algoritmo tradicional para o treinamento da MLP é o *backpropagation*. Esse algoritmo possui uma implementação relativamente simples e funciona bem para maioria dos problemas. Porém, ele possui algumas desvantagens como, por exemplo:

- Possui baixa taxa de convergência o que requer um tempo longo para o treinamento;
- Ocasionalmente ele pode ficar preso em mínimos locais;
- Trabalha com a minimização de apenas uma função objetivo (geralmente o erro médio quadrático);
- Possui dois parâmetros (taxa de aprendizado e *momentum*) que, às vezes, a depender do problema, pode dificultar sua utilização para um determinado usuário.

Neste trabalho, é apresentado um novo algoritmo, inspirado nos SIAs, para o problema do **treinamento de uma rede MLP**.

1.2 Objetivos

O principal objetivo deste trabalho é propor um novo algoritmo (iMaRag) para ser utilizado no treinamento de uma rede MLP. Na literatura, diversos autores propuseram abordagens com algoritmos bio-inspirados (algoritmos genéticos, otimização por enxames de partículas entre outros) (QIANG Gao *et al*, 2005; CARVALHO, Marcio Ribeiro de, 2007; SLOWIK, A., BIALKO, M., 2008) para este tipo de problema. Entretanto, uma abordagem baseada em sistemas imunológicos ainda não foi realizada para este tipo de problema e este projeto é uma iniciativa pioneira. Além disso, os outros objetivos estão relacionados a melhorar os aspectos de: velocidade de convergência no treinamento, garantir que o algoritmo não fique preso em mínimos locais e automatizar os parâmetros de configuração. Um dos objetivos para trabalhos futuros é acrescentar a característica multi-objetivo no algoritmo, possibilitando a minimização de mais de uma função.

1.3 Estrutura da monografia

Este trabalho está dividido em 6 capítulos. No primeiro capítulo, foi apresentada uma breve introdução ao tema do projeto e também a definição dos problemas e os objetivos para a pesquisa.

No capítulo 2 é feita uma explanação sobre RNAs, definindo suas origens e mostrando suas principais características e aplicações.

No capítulo 3, são discutidos os SIAs, suas semelhanças com o sistema imunológico biológico e seus principais modelos, mecanismos e funcionalidades.

No capítulo 4, é apresentada a metodologia utilizada para a construção do algoritmo proposto. Aqui também são discutidas as principais mudanças em relação ao modelo escolhido e as razões para essas mudanças.

No capítulo 5, são ilustrados os principais resultados alcançados e uma comparação do iMaRag com o *backpropagation* para algumas bases conhecidas da literatura.

No capítulo 6, uma conclusão é apresentada juntamente com as perspectivas futuras para o algoritmo.

2. Redes Neurais Artificiais (RNAs)

Este capítulo tem como objetivo fundamentar a técnica de computação inteligente (EVANGELIA MICHELI-TZANAKOU, 1997) conhecida como Rede Neural Artificial (RNA), que foi utilizada no trabalho. Mais precisamente, foi utilizada uma RNA do tipo MLP (*Multi-Layer Perceptron*) que também será explicada em detalhes.

As RNA's, também conhecidas como métodos conexionistas, são técnicas de computação inteligente que se baseiam na organização e no funcionamento do cérebro humano (BRAGA, Antônio, CARVALHO, André e LUDEMIR, Teresa, 2000; OSÓRIO, Fernando, 1999). A RNA também pode ser vista como um processador paralelo e distribuído que é formado por unidades de processamentos mais simples e que possui capacidade para armazenar conhecimento experimental e torná-lo disponível para utilização (HAYKIN, Simon, 2001).

Com isso em mente, podemos dizer que as RNAs emulam o funcionamento do cérebro humano e possuem duas características fundamentais: capacidade de aprendizado e armazenamento do conhecimento para generalização. O cérebro humano é formado por células especializadas chamadas neurônios que estão interconectadas entre si, formando uma imensa rede neural biológica. Para realizar o processamento da informação, cada neurônio recebe sinais de outros neurônios, combina estes sinais, realiza um processamento interno e então envia sinais a um grande número de outros neurônios. A capacidade de aprendizado e conseqüente generalização do cérebro é função do padrão de conexões entre os neurônios. Então, para o melhor entendimento e compreensão do funcionamento de uma RNA, é interessante um breve estudo sobre o funcionamento dessas células específicas do cérebro humano conhecidas como neurônios.

2.1 Modelo biológico do neurônio

O neurônio (Figura 1) é uma célula nervosa e a unidade básica do sistema nervoso humano. Eles fazem o trabalho de comunicação com o cérebro. Cada neurônio é capaz de transportar sinais eletroquímicos, bem como de retransmitir os sinais para outros neurônios.

O neurônio tem quatro regiões principais em sua estrutura. O corpo celular, os dendritos, o axônio e as terminações do axônio. O corpo celular é o “coração” do neurônio, ele contém o núcleo celular e mantém a síntese de proteínas dessa célula. O neurônio pode ter vários dendritos, os quais lembram galhos de uma árvore e são responsáveis pelo recebimento de sinais que chegam através de outros neurônios. O axônio é responsável por conduzir, através de toda sua extensão, os sinais elétricos gerados pelo neurônio ou transmitidos para o neurônio. No final do axônio, existem suas terminações, que são vários braços e que funcionam como terminais pré-sinápticos. Então, os neurônios recebem sinais de outros neurônios e se comunicam através de um processo denominado de sinapse (Figura 2). Sinapse é a região onde dois neurônios entram em contato e através da qual os impulsos nervosos são transmitidos entre eles. Em seguida esse sinal (o qual podemos chamar de informação) é processado no corpo celular e após isso, propagado através do axônio (KULKARNI, Arun D, 2001) até os terminais axônicos. Porém, essa informação só será novamente transmitida para outro neurônio se ela atingir um limiar excitatório (Lei do Tudo ou Nada). Um neurônio visto isoladamente, apenas compreende mais uma célula do corpo humano. O grande diferencial destas células é essa capacidade de comunicação descrita, com a propagação de impulsos nervosos (sinais elétricos ou informações) formando uma grande rede de neurônios. Em estudo recente (AZEVEDO, Frederico A.C. *et al*, 2009) foi calculado que um homem sadio entre 50 e 70 anos possui 86 bilhões de neurônios onde cada neurônio forma, em média, mil a dez mil sinapses o que remete a formação de redes neurais extremamente complexas.

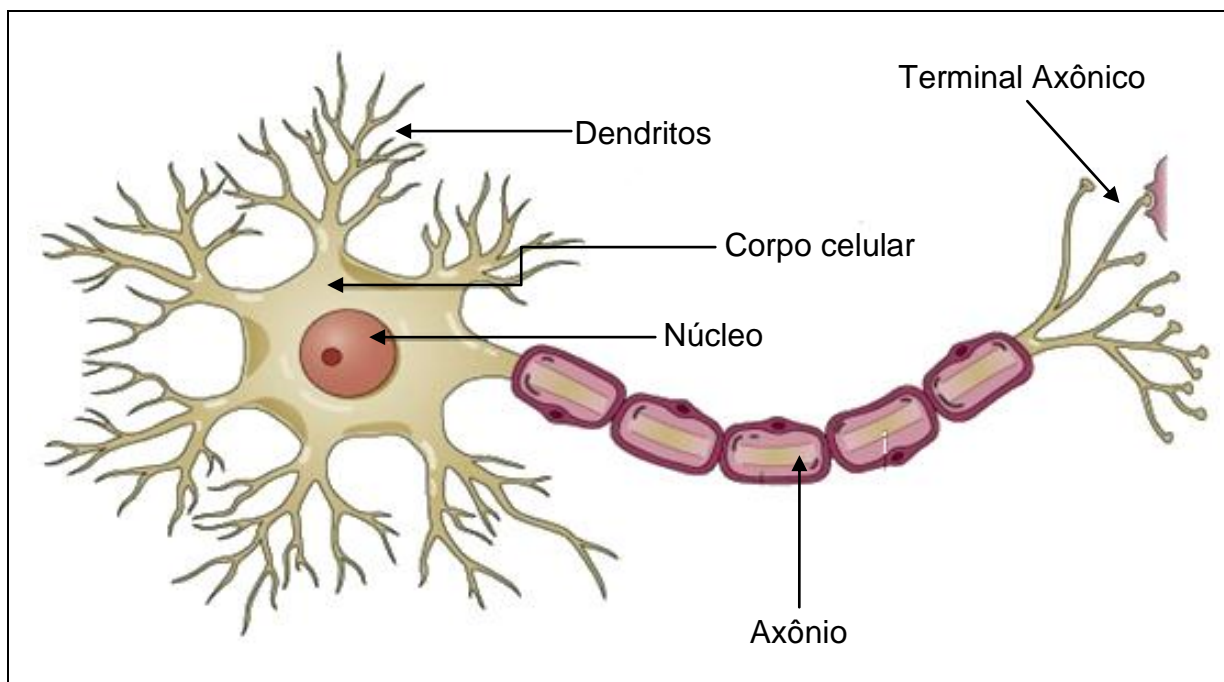


Figura 1. Neurônio e seus componentes (SCIENCEBLOGS)

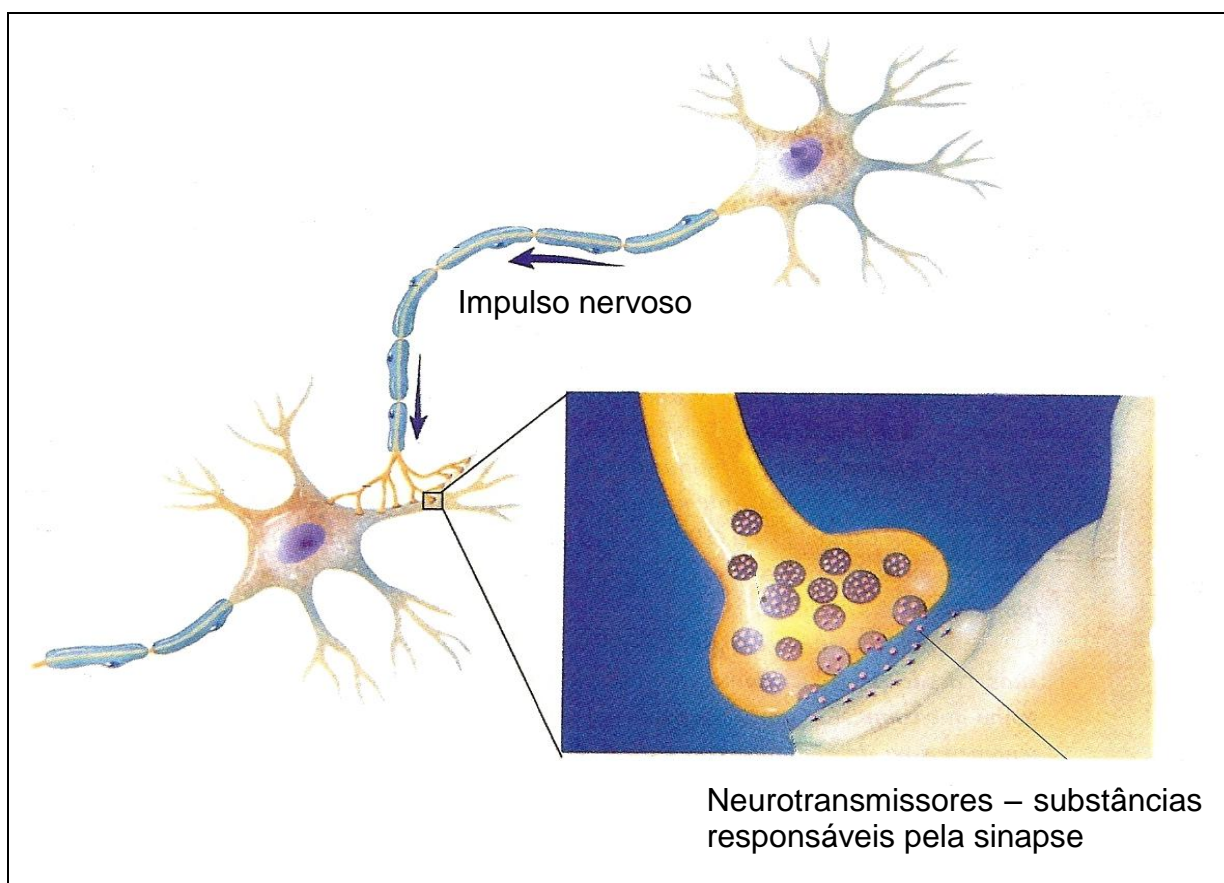


Figura 2. Sinapse entre dois neurônios (CLARINDASOUSA)

2.2 Modelo computacional do neurônio

O primeiro modelo computacional de um neurônio foi proposto por McCulloch e Pitts (MCCULLOCH, W. S. e PITTS, W., 1943) e é bastante simples mas de fundamental contribuição para o desenvolvimento das RNAs. Esse modelo consiste de um conjunto de entradas, uma unidade central de processamento e uma ou mais unidades de saída que representam, respectivamente, os dendritos, o corpo celular e o terminal axônico. A maior limitação do modelo de neurônio de McCulloch e Pitts (MCP) é sua natureza binária, tanto para as entradas como as saídas. O funcionamento do modelo pode ser descrito intuitivamente da seguinte maneira: se a soma ponderada dos sinais de entrada de um neurônio ultrapassar um determinado limiar, então a saída assume valor um (1) caso contrário, assume o valor zero (0). Para realizar isso, MCP utilizaram uma regra de propagação e uma função de ativação. A emissão de sinal por este neurônio está baseada em um limiar em analogia com a Lei do Tudo ou Nada (VALENÇA, Mêuser, 2009).

Considere $x_1, x_2, x_3, \dots, x_n$, como sendo as variáveis de entrada x_j ($j = 1, \dots, n$) do neurônio de saída 'i'. A entrada líquida net_i é dada pela seguinte regra de propagação:

$$net_i = \sum_{j=1}^n w_{ij} \times x_j - \theta \quad (1)$$

Onde: w_{ij} são os pesos sinápticos e θ é o limiar.

A Figura 3 mostra a representação do neurônio de MCP.

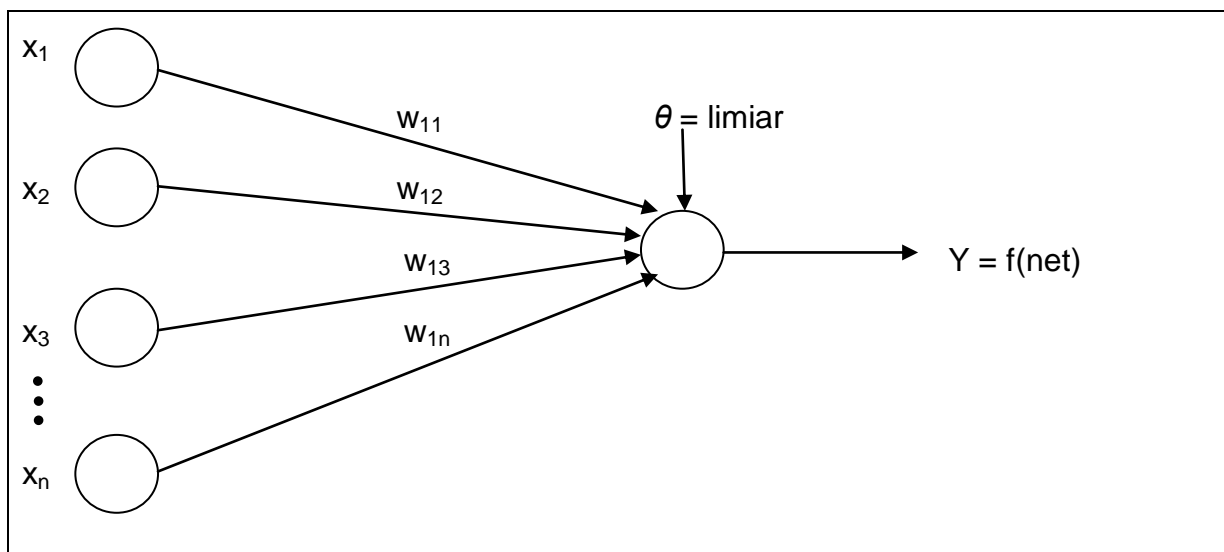


Figura 3. Neurônio de MCP com limiar explícito (VALENÇA, Mêuser, 2009).

Para a modelagem de MCP a função de ativação $f(net)$ é a função degrau, isto é:

$$f(net_i) = \begin{cases} 1, \forall net_i \geq 0 \\ 0, \forall net_i < 0 \end{cases} \quad (2)$$

Porém, nos algoritmos de aprendizagem, utiliza-se a representação com o limiar implícito. Para obter essa representação é necessário acrescentar uma nova entrada x_0 de valor fixo igual a um (1) e um novo peso sináptico $w_{10} = -\theta$ (limiar). Com essas modificações temos uma nova representação, condizente com as representações utilizadas na literatura atual (Figura 4) e um novo valor para cálculo do net_i .

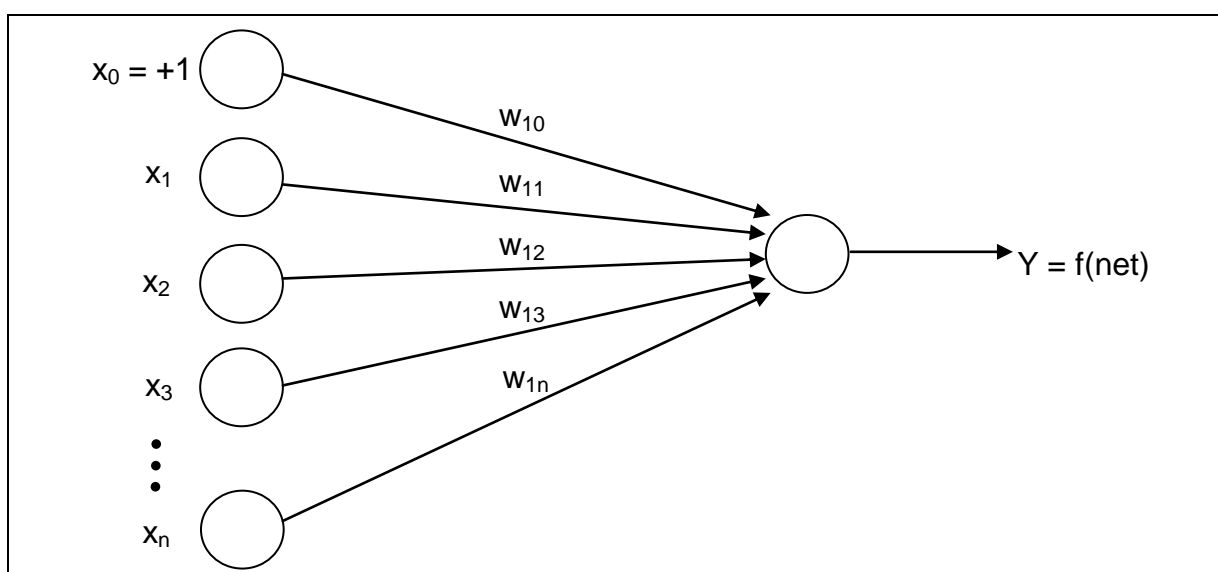


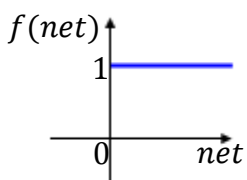
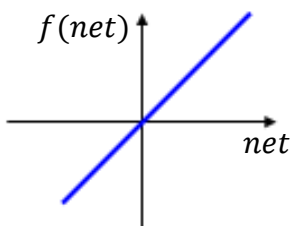
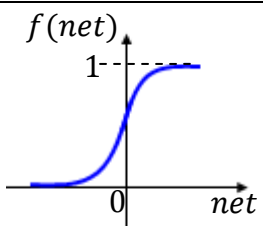
Figura 4. Neurônio de MCP com limiar implícito (VALENÇA, Mêuser, 2009).

Com essa representação, o cálculo do net_i fica da seguinte maneira:

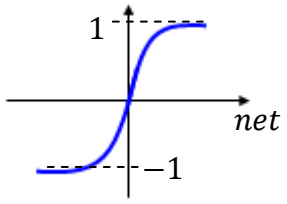
$$net_i = \sum_{j=0}^n w_{ij} \times x_j \quad (3)$$

O peso sináptico é um fator que pondera o valor de entrada de acordo com a importância desta entrada para obtenção da saída esperada. Já a função de saída, ou função de ativação $f(net)$ depende muito do problema proposto, tanto pode ser discreta como contínua como também pode ser linear ou não-linear. A consequência disso é que a RNA torna-se capaz de resolver vários problemas complexos do mundo real. A tabela 1 ilustra a equação e o gráfico de algumas funções de ativação que são mais utilizadas na literatura.

Tabela 1. Alguns exemplos de funções de ativação

Função	Equação	Gráfico
Degrau	$y = f(net) = \begin{cases} 1, \forall net \geq 0 \\ 0, \forall net < 0 \end{cases}$	
Linear	$y = f(net) = net$	
Sigmoidal Logística	$y = f(net) = \frac{1}{1 + e^{-net}}$	

$f(net)$

Tangente Hiperbólica	$y = f(net) = \frac{e^{net} - e^{-net}}{e^{net} + e^{-net}}$	
----------------------	--	---

2.3 Regra de Aprendizado

A característica que torna a RNA tão poderosa é sua capacidade de aprender através de exemplos e, dessa maneira, generalizar a informação aprendida. (BRAGA, Antônio, CARVALHO, André e LUDEMIR, Teresa, 2000). Portanto, a propriedade mais importante da RNA é o seu aprendizado, de acordo com os exemplos mostrados (geralmente provenientes de bases de dados construídas por especialistas), a rede extrai o conhecimento e aplica este conhecimento para encontrar a solução de novos padrões. O aprendizado, ou treinamento de uma RNA, é um processo iterativo, no qual os pesos vão sendo ajustados de maneira gradual. Diz-se que uma RNA está ajustada quando o treinamento chega ao fim e os pesos encontrados foram os melhores durante o ajustamento no aprendizado. Como já foi explicado anteriormente, o peso da RNA é um valor decimal que pondera o valor de entrada de acordo com a importância desta entrada para o problema em questão. O tipo de aprendizagem é determinado pela maneira que a modificação dos pesos sinápticos ocorre (MENDEL, J.M., MCLAREN, R.W., 1970). Existem diversas regras ou processos de aprendizagem (HAYKIN, Simon, 2001), porém, neste trabalho, vamos voltar a atenção para dois paradigmas de aprendizagem que são muito utilizados na literatura: o aprendizado supervisionado e o aprendizado não-supervisionado.

2.3.1 Aprendizado com um Supervisor

O aprendizado com um supervisor ou aprendizado supervisionado é uma forma de aprendizagem de máquina que procura deduzir uma função a partir de um conjunto de dados de treinamento. O conjunto de dados de treinamento consiste em pares de entrada e suas respectivas saídas esperadas, estas saídas são conhecidas pelo supervisor. Dessa maneira, a(s) entrada(s) é/são disponibilizada(s) para a RNA, que faz o processamento e calcula a saída, nesse momento entra o papel do

supervisor, que compara a saída desejada (que ele já conhece para esse conjunto de entradas) com a saída calculada pela RNA, computa o erro e o retorna para a RNA. Esse retorno do erro para a RNA é chamado de *feedback* e ele é utilizado para a RNA fazer um ajuste nos pesos visando minimizar o erro a cada ciclo de treinamento. A Figura 5 demonstra graficamente o processo de aprendizagem supervisionado.

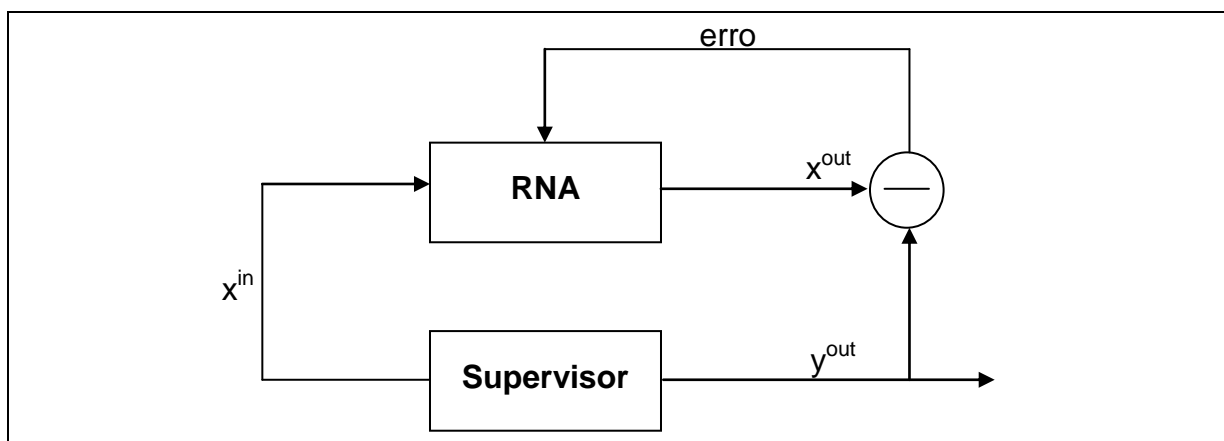


Figura 5. Esquema de aprendizado supervisionado.

Na figura acima, x^{in} significa a entrada que o supervisor disponibiliza para a RNA, x^{out} significa a saída calculada pela rede e y^{out} é a saída esperada para essa entrada. O erro é calculado subtraindo a saída desejada pela saída calculada: $erro = y^{out} - x^{out}$. Por fim, o erro alimenta a RNA para o ajuste dos pesos sinápticos. O aprendizado supervisionado é amplamente utilizado nas redes do tipo MLP, que serão detalhadas na seção 2.4.

2.3.2 **Aprendizado sem um Supervisor**

Este tipo de paradigma geralmente é subdividido em: aprendizagem por reforço e aprendizagem não-supervisionada.

Aprendizagem por reforço

Na aprendizagem por reforço, é realizada uma interação contínua com o conjunto de entradas e saídas visando minimizar um índice escalar de desempenho. É uma forma de aprendizagem que é formulada em torno de um *crítico* que converte um sinal de reforço primário recebido do ambiente em que se encontra a rede em um sinal de reforço heurístico, que possui uma melhor qualidade em relação ao primário. Ambos os sinais são entradas escalares (BARTO ET AL., 1983). Ações são

tomadas seqüencialmente ao longo do tempo e o objetivo é minimizar uma função de custo cumulativa e descobrir que ações resultaram em um melhor comportamento global do sistema.

Aprendizagem não-supervisionada

A aprendizagem não-supervisionada, também conhecida como auto-organizada, como o próprio nome diz, não utiliza um supervisor que disponibiliza as saídas desejadas para cada conjunto de entradas nem um crítico para supervisionar o processo. Dessa forma, nesse paradigma de aprendizagem, utilizam-se apenas os valores dos dados de entrada e há uma tentativa de encontrar padrões para criar representações internas de acordo com as características desses dados (BECKER, S., 1991). Dentre as redes que utilizam o aprendizado não-supervisionado destacam-se os SOMs (*Self-Organization maps*), ou em português: Mapas Auto-Organizáveis (KOHONEN, Teuvo, 1982).

2.4 Multi-Layer Perceptron (MLP)

2.4.1 Perceptron

Antes de entrar em detalhes sobre uma rede MLP, é interessante explanar a rede *perceptron*, que foi a precursora da MLP. O *perceptron*, proposto por Frank Rosenblatt em 1958 (ROSENBLATT, Frank, 1958), é o modelo mais simples de uma RNA. Ele é formado por várias unidades de processamento que são interligadas através de conexões, que são os pesos sinápticos. O *perceptron* utiliza a formulação do neurônio de MCP e, portanto a sua saída obedece a Lei do Tudo ou Nada. Dessa maneira a saída do *perceptron* é determinada por uma função degrau que define quando o neurônio está ativo (saída 1) ou em repouso (saída 0). A Figura 6 apresenta uma arquitetura genérica para o *perceptron*. Com a sua simplicidade o *perceptron* carrega uma grande limitação: ele só é capaz de resolver problemas que sejam linearmente separáveis. Entende-se por problemas linearmente separáveis, os problemas cuja solução pode ser obtida pela separação de duas regiões por meio de uma reta (Figura 7). Como a maioria dos problemas reais não são linearmente separáveis, é justificável uma melhoria sobre o *perceptron*.

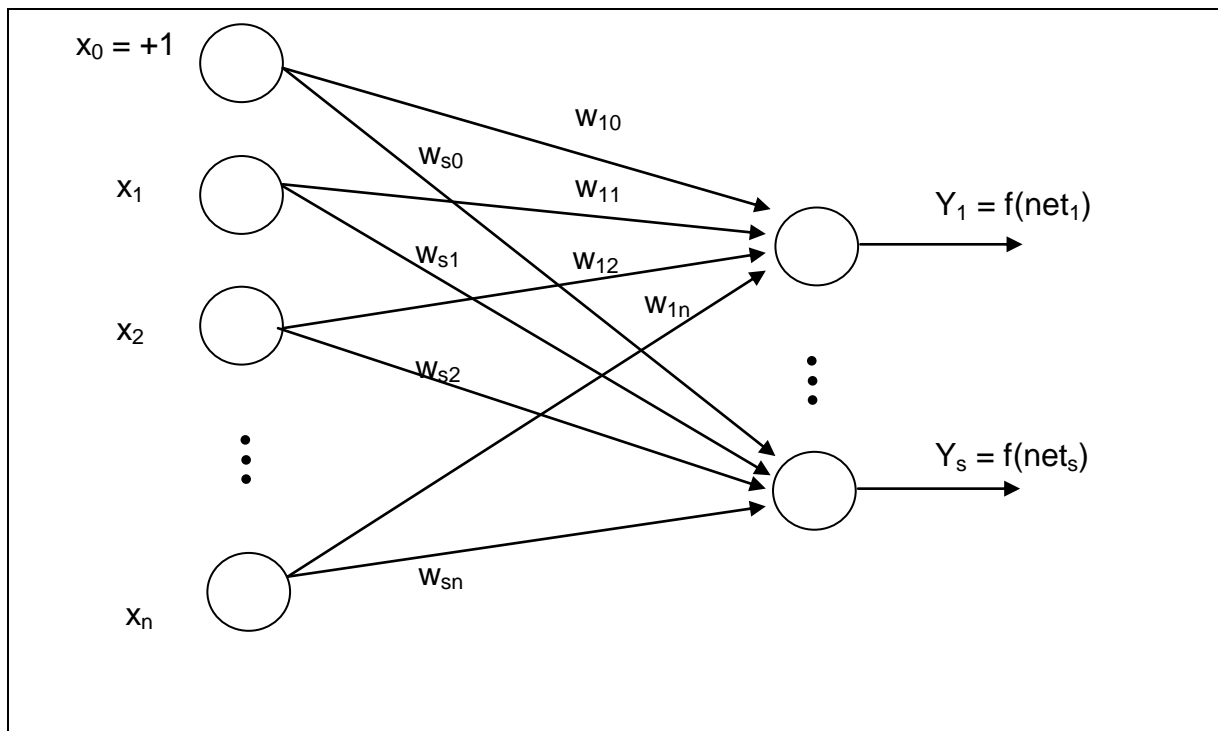


Figura 6. Arquitetura do perceptron

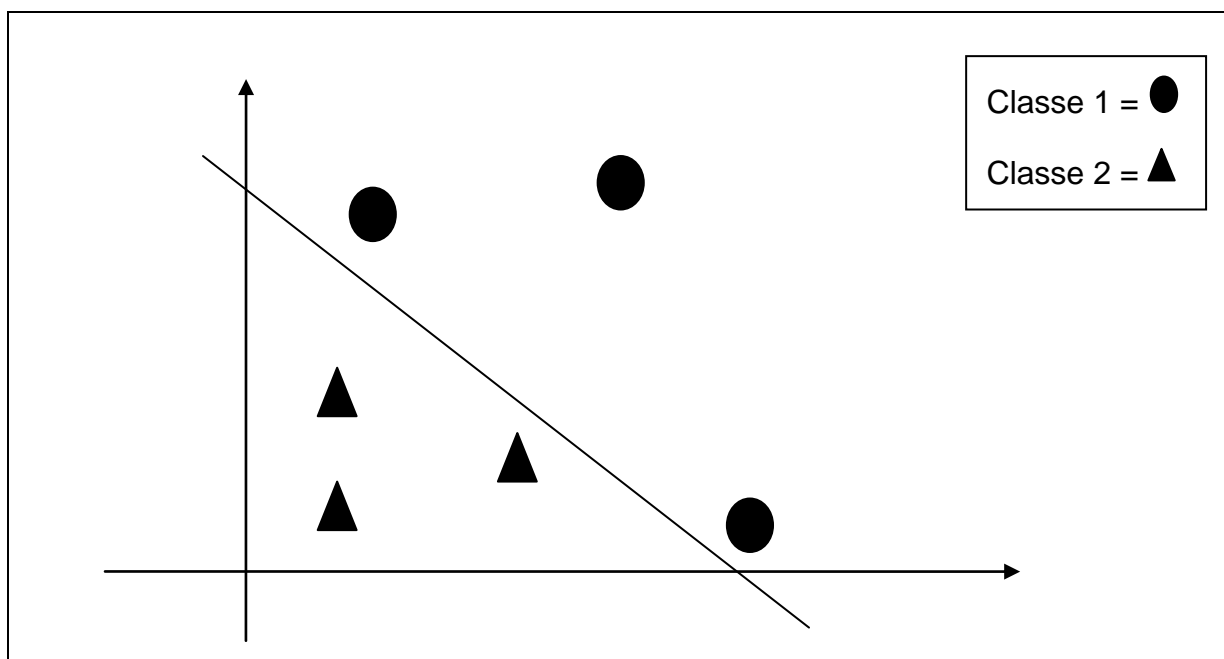


Figura 7. Exemplo de problema linearmente separável

2.4.2 Arquitetura

Uma rede *perceptron* de múltiplas camadas (MLP), com uma função de ativação não linear na camada escondida, representa uma generalização do *perceptron*, ou seja, é formada por múltiplas camadas de neurônios *perceptron*. Uma

rede MLP é formada por uma camada de entrada, uma ou mais camadas intermediárias (também chamadas de camadas escondidas/ocultas) e uma camada de saída. A utilização de uma camada intermediária, com função de ativação não linear, tal qual a sigmóide logística (Tabela 1), aumenta o poder computacional das redes MLP (BRAGA, Antônio, CARVALHO, André e LUDEMIR, Teresa, 2000) e torna possível para a rede a aproximação de qualquer função contínua (CYBENKO, George, 1989). Já a utilização de duas camadas intermediárias permite a aproximação de qualquer função matemática (CYBENKO, George, 1988). Por essa razão, as redes MLP são conhecidas como uma técnica aproximadora universal de funções. A Figura 8 ilustra a arquitetura de uma rede MLP com duas camadas intermediárias (HAYKIN, Simon, 2001).

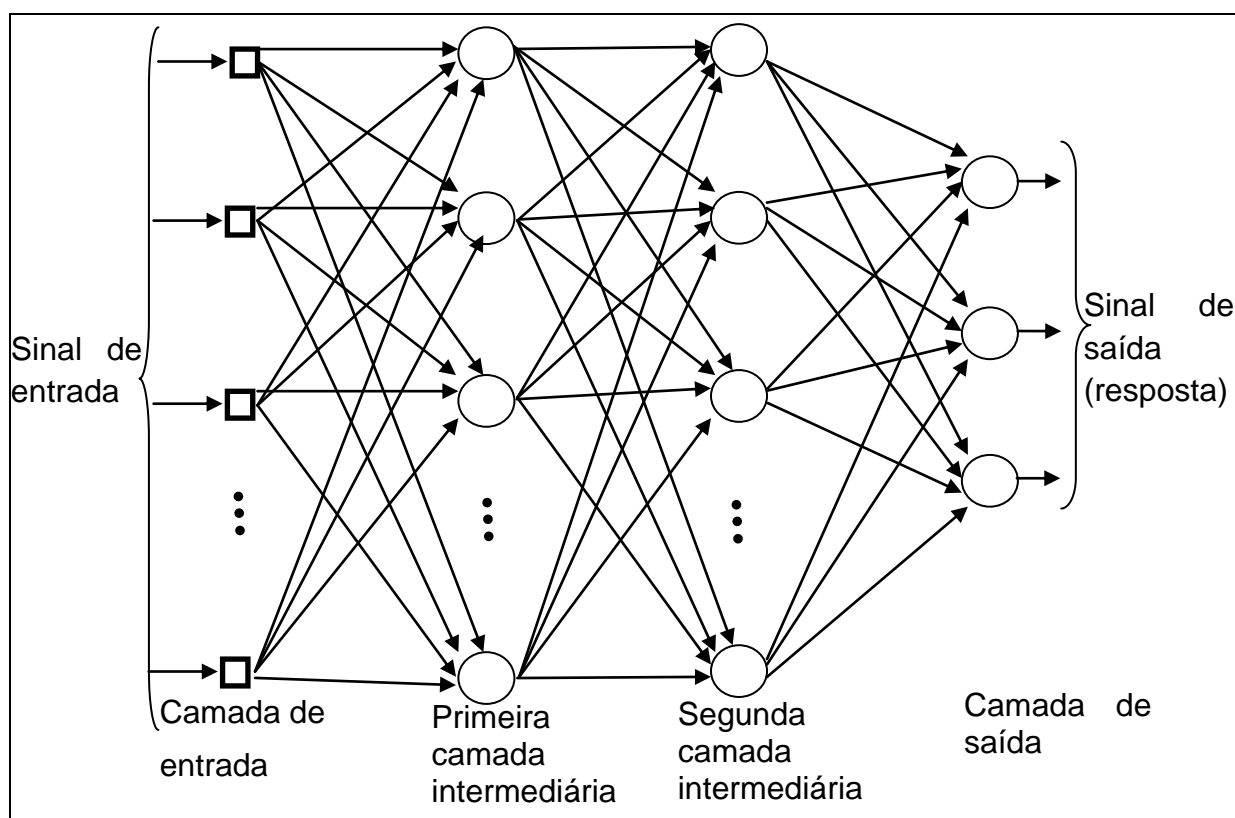


Figura 8. Arquitetura de uma rede MLP com duas camadas intermediárias

2.4.3 Características

- A camada de entrada de uma rede MLP possui apenas a característica de obtenção dos dados, não realiza qualquer tipo de processamento;
- A rede possui no mínimo uma camada intermediária;

- A camada de saída processa as saídas dos neurônios da camada anterior, que será, obrigatoriamente, uma camada intermediária;
- Cada neurônio da rede possui uma função de ativação (ver exemplos de função de ativação na Tabela 1), que geralmente é a sigmóide logística;
- A rede não possui conexões recorrentes, ou seja, não existem conexões entre neurônios de uma camada com os neurônios da camada anterior.

Além disso, é importante frisar que o número de camadas intermediárias, a quantidade de neurônios em cada camada, a função de ativação utilizada, a semente para inicialização aleatória dos pesos, entre outros, são fatores importantes para a arquitetura da rede, e que influenciarão no resultado final (BRAGA, Antônio, CARVALHO, André e LUDEMIR, Teresa, 2000).

Existe ainda uma grande dificuldade neste tipo de rede: o treinamento. Efetuar a correção dos pesos dos neurônios que se encontram na camada de entrada e na camada intermediária não é uma tarefa trivial (BRAGA, Antônio, CARVALHO, André e LUDEMIR, Teresa, 2000). Neste contexto, existem vários algoritmos para treinar uma rede neural (FAHLMAN, Scott, 1988; RIEDMILLER, Martin, 1994; HAGAN, Martin e MENHAJ, Mohammad, 1994). Neste trabalho, além do algoritmo baseado em SIA que é proposto, foi utilizado também o algoritmo *backpropagation*.

2.5 Algoritmo Backpropagation

O algoritmo *backpropagation* (NIELSEN, R., 1989) é o mais conhecido, e sem dúvida o mais utilizado da literatura para treinamento de uma rede neural. Ele é um algoritmo supervisionado que busca em seu processo de aprendizagem a minimização de uma função objetivo, usualmente o erro médio quadrático (EMQ), através do método do gradiente descendente. O *backpropagation* é composto por duas fases onde cada fase percorre a rede em um sentido:

2.5.1 Fase forward

A fase chamada de forward corresponde a fase de propagação progressiva do sinal, ou seja, o conjunto de sinais de entrada flui através da rede, da esquerda para direita ou da camada de entrada até a camada de saída, onde, no fim se

calcula a saída da rede para esse conjunto de sinais de entrada apresentado à rede (BRAGA, Antônio, CARVALHO, André e LUDEMIR, Teresa, 2000).

2.5.2 Fase backward

A fase chamada de backward corresponde à retropropagação do erro, onde a saída desejada da rede e a saída calculada da rede são utilizadas para o ajuste dos pesos das conexões (BRAGA, Antônio, CARVALHO, André e LUDEMIR, Teresa, 2000). Como o aprendizado é supervisionado, o intuito do algoritmo é ajustar os pesos da rede para que os erros de saída (desejada e calculada) seja minimizado.

A Figura 9 ilustra a porção de uma rede MLP e as duas fases do algoritmo *backpropagation*.

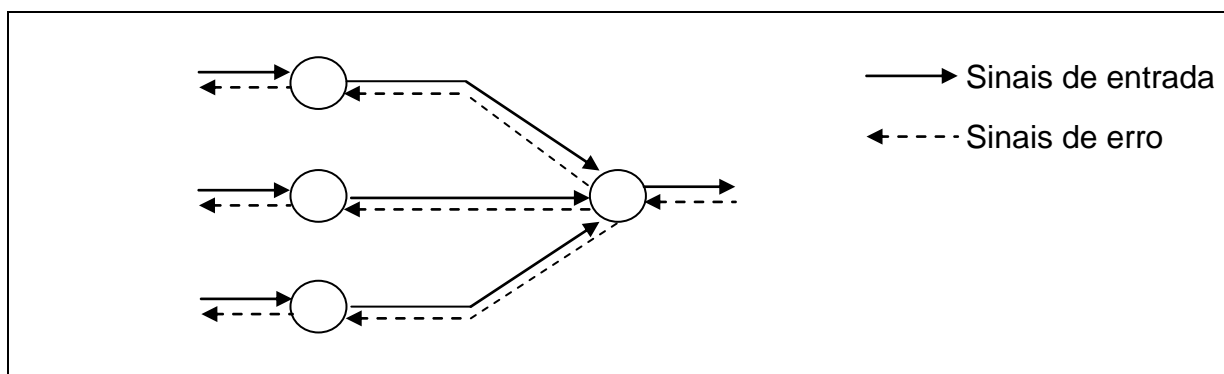


Figura 9. Fluxo do *backpropagation*: propagação para frente dos sinais de entrada e retropropagação de sinais de erro (HAYKIN, Simon, 2001)

Cada exemplo do conjunto de dados disponível para o treinamento é submetido às duas fases do *backpropagation*. Quando todos os exemplos são mostrados a rede, dizemos que foi completado um ciclo de treinamento e neste momento também é calculado o EMQ deste ciclo. Tipicamente esse EMQ decresce à medida que o número de ciclos avança, em outras palavras, o EMQ começa com um valor alto, decresce rapidamente e continua diminuindo lentamente conforme a rede segue seu caminho em direção a um mínimo local na superfície de erro (HAYKIN, Simon, 2001). Dessa maneira, entramos no problema do critério de parada do treinamento, ou seja, quando devemos interromper o treinamento e estimar que a rede esteja devidamente treinada? Uma alternativa simples seria estabelecer um valor de 'n' ciclos e que ao alcançar esse valor o treinamento seja encerrado. Porém esta abordagem simplista carrega o inconveniente de determinar

um bom valor para o número de ciclos, pois um valor pequeno pode ser insuficiente para a convergência da rede (*underfitting*) enquanto um valor alto pode levar à perda da capacidade de generalização da rede (*overfitting*). Além dessa abordagem simplista, existem outras abordagens para o critério de parada de treinamento de uma rede MLP, porém, neste trabalho iremos nos concentrar em apenas uma, que foi a técnica utilizada na implementação do trabalho: a validação cruzada.

2.6 Validação Cruzada (VC)

A técnica de VC é uma ferramenta padrão da estatística (Stone, M., 1974, 1978) que é utilizada, no contexto das RNAs, para impedir o *overfitting* (treinamento em excesso) da rede. Uma vez que uma rede é treinada em excesso, ela tende a memorizar ou decorar os exemplos de treinamento que são mostrados a ela e acaba perdendo a sua capacidade de generalização. Nesta técnica, o conjunto de dados disponível é dividido aleatoriamente em um conjunto de treinamento e outro de testes. Por sua vez, o conjunto de treinamento é dividido em dois subconjuntos: o subconjunto de estimação, que é o subconjunto utilizado no treinamento da rede, e o subconjunto de validação, que é o subconjunto utilizado na validação cruzada (HAYKIN, Simon, 2001). No capítulo 4 entraremos em mais detalhes sobre a divisão do conjunto de dados efetuada no trabalho.

A VC é realizada sempre que se conclui um ciclo de treinamento da rede, isto significa que a cada ciclo, após o reajuste dos pesos o treinamento é interrompido para o cálculo do Erro Médio Quadrático de Validação Cruzada (EMQVC) para esse ciclo. Neste momento, o subconjunto de dados específico para a VC será submetido à rede para o cálculo do EMQVC. Como já foi explicado anteriormente, o EMQ de treinamento tende a decrescer à medida que os ciclos aumentam e para a VC assume-se que enquanto o EMQVC continue decrescendo a cada ciclo, a rede continua generalizando. Caso o EMQVC comece a crescer por um número de ciclos, significa que a rede não está mais generalizando e o treinamento deve ser interrompido. Neste caso, os pesos ajustados que devem ser utilizados para os testes posteriores são os pesos exatamente do ciclo anterior ao ciclo que marca a contínua ascensão do EMQVC. A figura abaixo demonstra o momento em que o treinamento deve ser interrompido para não haver perda de generalização.

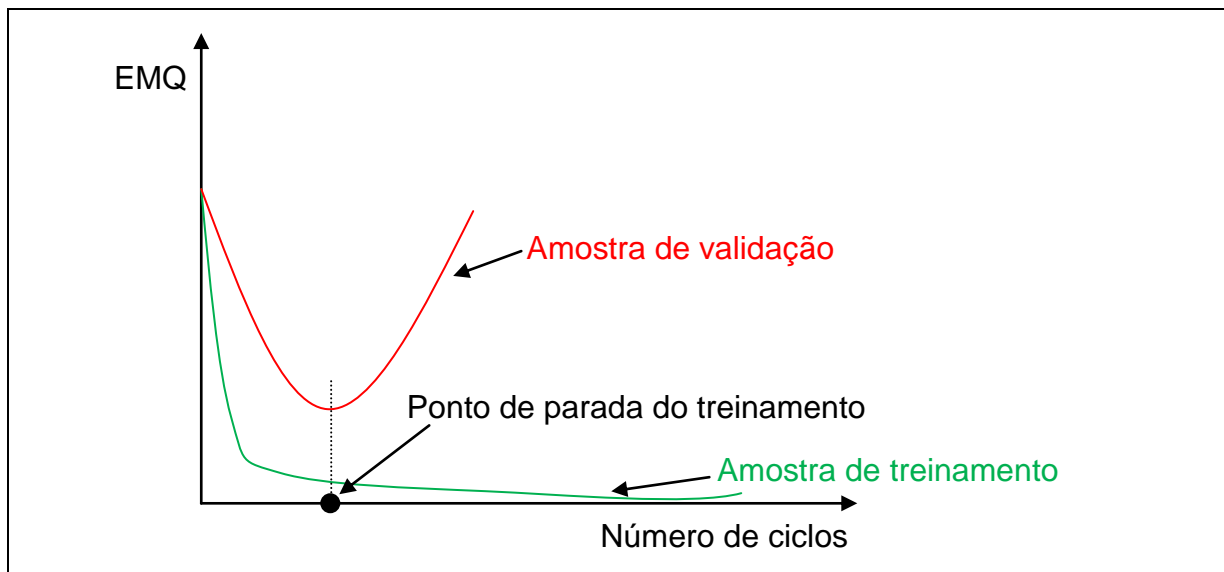


Figura 10. Critério de parada do treinamento baseado na VC

3. Sistemas Imunológicos

Este capítulo tem como objetivo fundamentar a técnica de Sistemas Imunológicos Artificiais (SIAs), que foi utilizada no trabalho. Na primeira parte, discute-se acerca do funcionamento e dos principais constituintes do Sistema Imunológico Biológico. Na segunda parte discute-se sobre os SIAs e algumas de suas aplicações inclusive a aplicação para qual ele foi desenvolvido neste trabalho. A discussão geral deste capítulo foi inspirada em DE CASTRO, L. N. (2001) e em BERBERT, P. C. (2008).

3.1 Sistema Imunológico Biológico

A *imunologia* é o ramo da biologia responsável pelo estudo das reações de defesa que conferem resistência às doenças (KLEIN J., 1990). Dentro deste contexto, o sistema imunológico biológico é um sistema robusto, complexo e adaptativo formado por moléculas, células e órgãos, cujo principal objetivo é a defesa do organismo contra agentes patogênicos (organismos invasores). Ele é um sistema capaz de reconhecer e distinguir todas as células do nosso corpo, desde as células seguras até as células que são consideradas perigosas, seja por serem invasoras seja por serem alteradas devido a alguma anomalia. Este sistema conta ainda com uma eficaz rede de comunicação e mensagens químicas entre as células que fazem parte dele. Como pode ser visto na Figura 11, o sistema imunológico é estruturado em múltiplos níveis (camadas), cada qual com sua específica função e mecanismos de defesa.

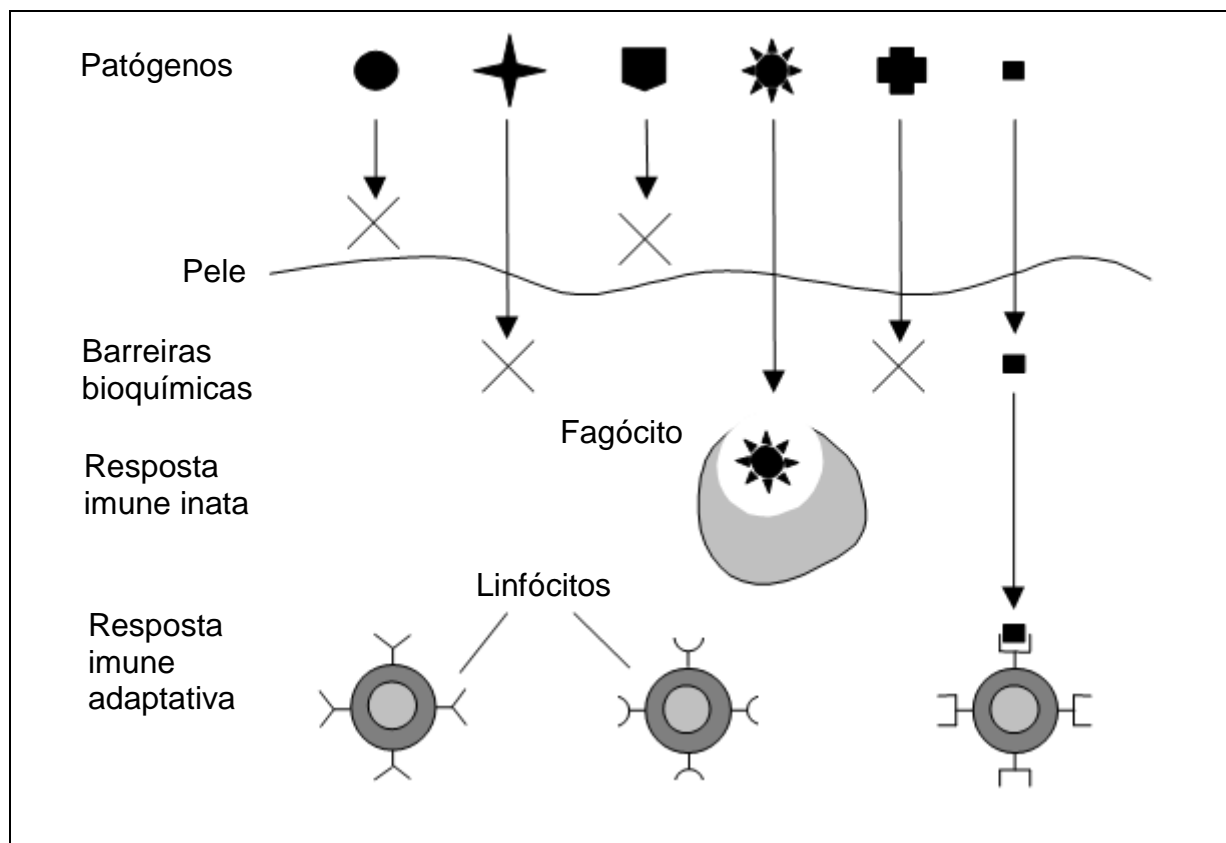


Figura 11. Arquitetura multicamadas do sistema imunológico (DE CASTRO, L. N., 2001)

- **Barreiras físicas:** A pele e o sistema respiratório agem como uma barreira na defesa contra os invasores. A pele é revestida, em seu estrato mais superficial, por uma camada de células mortas, impregnadas por queratina (uma proteína). Além de garantir certa impermeabilização, essa camada representa uma barreira mecânica eficaz contra a penetração de agentes infecciosos. Além disso, existem outros mecanismos de neutralização que funcionam basicamente na retenção de partículas nos pêlos (da pele e da parte interna do nariz) e no ato de expelir partículas via tosse e espirros.
- **Barreiras bioquímicas:** o suor, a saliva, a lágrima, muco nas vias respiratórias e digestivas, entre outros, também impedem ou mesmo eliminam alguns invasores. Seus principais mecanismos de defesa são as secreções, que contém algumas enzimas responsáveis pela eliminação das partículas invasoras. Esses dois tipos de barreiras (físicas e bioquímicas) buscam evitar a penetração do agente patogênico no organismo.

- Sistema imune inato: É a primeira linha defensiva do organismo após a invasão de um agente patogênico. No local da invasão, os tecidos lesados liberam determinadas substâncias que atuam como mensageiros químicos, informando ao restante do corpo o acontecido. Neste momento é que entram em ação as principais células que formam o sistema imune inato: as células fagocitárias. Os macrófagos e os neutrófilos são células fagocitárias, elas recebem essa denominação porque são capazes de fagocitar (englobar) os antígenos (agentes invasores) com o objetivo de eliminá-los. Caso o invasor ainda não tiver sido eliminado, o último, e mais eficiente nível de defesa é acionado: o sistema imune adaptativo. Vale ressaltar que o sistema imune inato, na maioria dos casos já é suficiente para conter os agentes patogênicos e que quando não o é, possui um papel fundamental na iniciação e no direcionamento da resposta imune adaptativa, principalmente porque esse último nível de defesa demora certo período de tempo (da ordem de dias) para exercer seus efeitos.
- Sistema imune adaptativo: É o último nível de defesa contra os organismos invasores. Basicamente, é formado por células chamadas de *linfócitos* e possui mecanismos eficientes de memória e diversidade para combate ao invasor. Esses mecanismos são a principal inspiração para a técnica dos sistemas imunológicos artificiais, por isso, o sistema imune adaptativo requer uma explicação mais apurada.

3.1.1 Sistema imune adaptativo

Os linfócitos são as principais células que compõem o sistema imune adaptativo. Eles são divididos em linfócitos T (por dependerem do “Timo” para o seu desenvolvimento) e linfócitos B (por ter sua origem descoberta na cloaca das aves, especificamente na “Bolsa de Fabricius”) e ambos são capazes de reconhecer de forma específica os antígenos, distinguindo-os dos componentes do próprio organismo. Eles conseguem reconhecer os antígenos de forma específica através de seus *receptores*, cada linfócito entre os milhares presentes no organismo, possui um receptor específico e diferente. Mas, se um antígeno invasor possuir uma “forma” que não é conhecida de nenhum receptor dos milhares de linfócitos do organismo, como ele consegue eliminar esse invasor? Acontece que os linfócitos sofrem um tipo

de processo semelhante ao processo de *seleção natural* proposto por Charles Darwin: nesse caso, somente os linfócitos que encontrarem um antígeno no qual seu receptor possa “encaixar” serão ativados para proliferar e sofrer um processo de diferenciação em células efetoras e células de memória.

Dessa maneira, o linfócito que possuir um receptor que encaixe no antígeno é ativado e passa por um processo de proliferação por clonagem, onde clones do linfócito são originados e em seguida esses clones sofrem um processo de diversidade (ou diferenciação) onde uma porção deles irá se diferenciar em *plasmócitos* (células efetoras) que secretam os *anticorpos* com a mesma especificidade do receptor e outra porção irá se diferenciar em células de memória que são células que ficam de prontidão para uma defesa mais rápida caso haja uma invasão pelo mesmo antígeno. Os anticorpos são as células que de fato, efetuam a defesa do organismo, sua atuação é específica para o antígeno contra o qual ele foi produzido. Por exemplo, o anticorpo específico para o vírus causador da caxumba não se liga ao vírus causador do sarampo, e vice-versa. A Figura 12 ilustra esse processo explicado acima, que recebeu o nome de Seleção Clonal e que constitui a parte central da imunidade adaptativa.

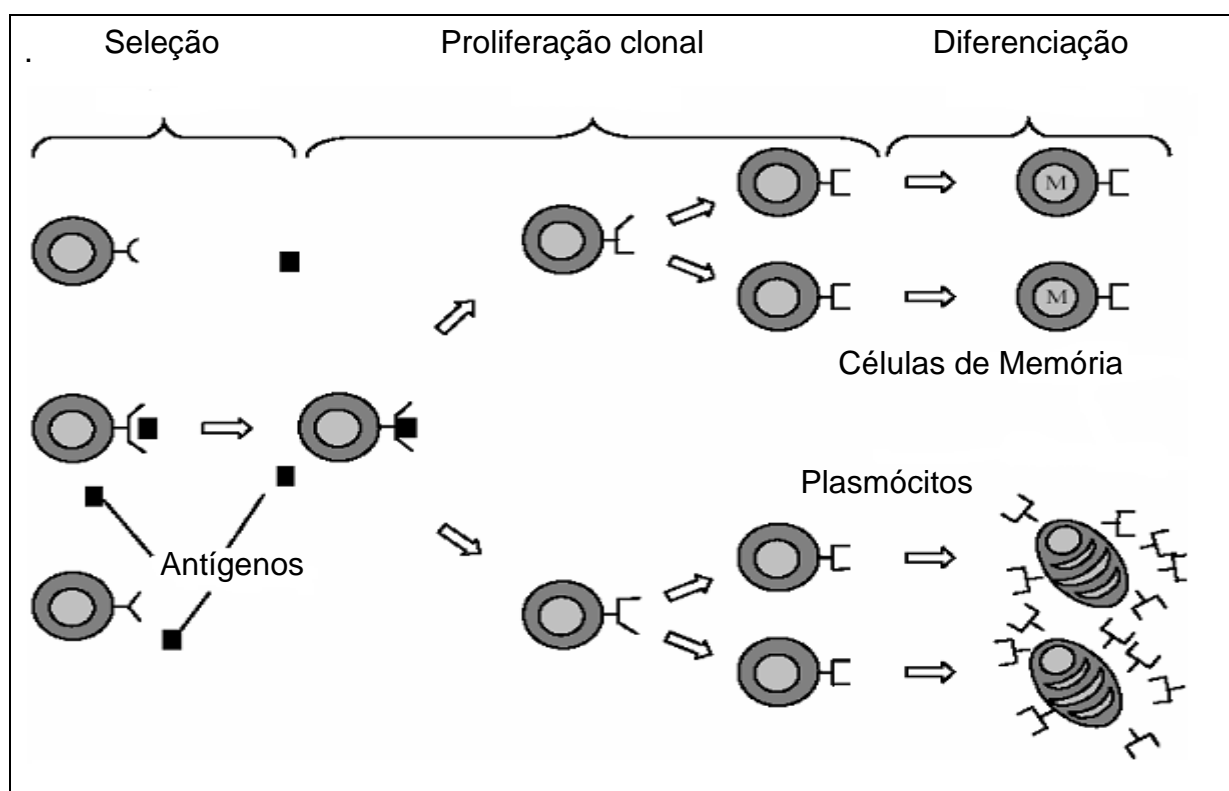


Figura 12. O sistema imune adaptativo (DE CASTRO, L. N., 2001)

Outro princípio importante é a contínua produção de novos anticorpos, processo que é chamado de metadinâmica. Existe uma constante renovação do repertório linfocitário, e conseqüentemente a morte de elementos que não são estimulados. A metadinâmica é outro processo do Sistema Imunológico que é capaz de introduzir diversidade, garantindo sua capacidade de combater novos antígenos (BERBERT, P. C., 2008).

Para explicar o sistema imunológico biológico de forma sucinta, foram omitidos vários detalhes biológicos e bioquímicos para não fugir do foco do trabalho, porém, vale ressaltar que esse sistema possui ainda uma grande variedade de componentes e funcionalidades resultando em um alto grau de complexidade. Na próxima seção iremos entrar em detalhes sobre os sistemas imunológicos artificiais, que se originaram a partir de simplificações dos mecanismos do sistema imunológico biológico.

3.2 Sistemas Imunológicos Artificiais (SIAs)

Assim como as RNAs, os Sistemas Imunológicos Artificiais são sistemas bio-inspirados, ou seja, se inspiram na natureza, na tentativa de simular mecanismos particulares e criar sistemas artificiais para a resolução de problemas complexos. Como pressuposto de qualquer sistema bio-inspirado, os diversos agentes operam em conjunto, e a conseqüência disso é que comportamentos complexos podem emergir. Os sistemas vivos são, sem sombra de dúvidas, *sistemas complexos* (DE CASTRO, L. N., 2001). Sistemas complexos podem ser tratados como uma coleção de elementos que exibem um comportamento dinâmico, coletivo e integrado, no qual a ação de um elemento possivelmente afeta as ações posteriores dos outros elementos, conglomerando sua interação com o ambiente.

A área de pesquisa dos Sistemas Imunológicos Artificiais é extensa. A escolha dos melhores modelos (Seleção Negativa, Teoria de Rede Imunológica, Seleção Clonal) depende do objetivo e das características do problema a ser estudado. As diferentes peculiaridades, funcionalidades, propriedades e mecanismos dos Sistemas Imunológicos Artificiais permitem sua aplicação em diversos problemas (Tabela 2).

Diante da quantidade de modelos e aplicações dos Sistemas Imunológicos Artificiais, não existe um esquema geral de quais elementos essenciais um algoritmo deve possuir, como é comum em outras técnicas bio-inspiradas (i.e.: algoritmos genéticos (AG), otimização por enxames de partículas (PSO)), que possuem um algoritmo geral único. A Figura 13 mostra os diversos modelos originados a partir dos SIAs.

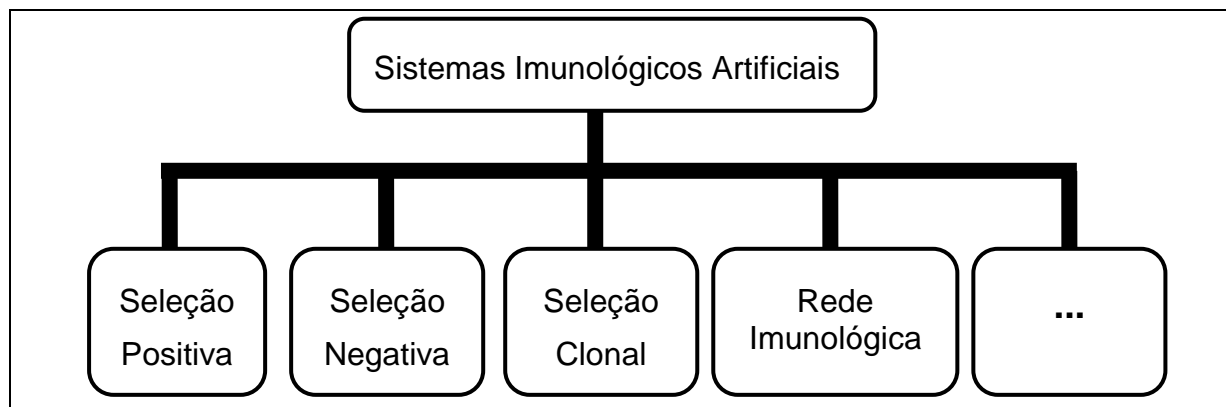


Figura 13. Exemplos de modelos dos SIAs

Segundo Dasgupta (1998) os Sistemas Imunológicos Artificiais são ferramentas computacionais compostos por metodologias inteligentes, inspirados no Sistema Imunológico Biológico, para a solução de problemas do mundo real. Portanto um SIA é um sistema computacional que se baseia nos princípios do sistema imunológico natural. Os SIAs surgiram a partir de tentativas de modelar e aplicar princípios imunológicos no desenvolvimento de novas ferramentas computacionais (Tabela 2): reconhecimento de padrões, detecção de faltas e anomalias, segurança computacional, busca e otimização, controle, robótica, *scheduling*, análise de dados, aprendizagem de máquina, processamento de informações, resolução de problemas, etc (BERBERT, P. C., 2008).

Tabela 2. Modelos de SIA e suas resoluções práticas (DASGUPTA, D., 2006)

Aspecto do Sistema Imunológico Biológico	Problema Computacional	Aplicações Típicas
Reconhecimento de próprio e não-próprio do organismo	Detecção de mudanças ou anomalias	✓ Segurança de computador ✓ Detecção de falta

Aspecto do Sistema Imunológico Biológico	Problema Computacional	Aplicações Típicas
Teoria de Rede Imunológica e Memória Imunológica	Aprendizagem (supervisionada ou não-supervisionada)	<ul style="list-style-type: none"> ✓ Classificação ✓ Clusterização ✓ Análise de dados ✓ Mineração de dados
Seleção Clonal	Busca, Otimização	<ul style="list-style-type: none"> ✓ Otimização de funções
Mobilidade e Distribuição	Processamento Distribuído	<ul style="list-style-type: none"> ✓ Arquiteturas de agente ✓ Controle robótico descentralizado
Imunidade Inata	Teoria do Perigo	<ul style="list-style-type: none"> ✓ Segurança de redes

O modelo de SIA selecionado para este trabalho é o de *seleção clonal* e na próxima seção serão abordados alguns detalhes sobre seus principais componentes e funcionalidades.

3.3 Seleção Clonal

Este modelo foi selecionado pela característica do problema a ser resolvido. Como já foi discutido, este trabalho propõe a utilização de um algoritmo baseado em SIAs, o iMaRag, para o treinamento de uma rede MLP. Dentro deste contexto, o algoritmo será utilizado para minimizar o EMQ buscando encontrar o conjunto de pesos (dentre aqueles que foram gerados inicialmente) que represente a melhor generalização para a rede.

Neste modelo, existe o conceito de *repertório de anticorpos*, no qual cada anticorpo representa uma solução possível para o problema em questão. Sendo assim, o repertório de anticorpos representa o conjunto de possíveis soluções para o problema. A cada anticorpo é atribuído um *fitness* que é definido a partir de critérios de dominância (explicado logo a seguir) e que representa o quão bom esse anticorpo é. Anticorpos que possuem os maiores *fitness* são os melhores anticorpos do repertório. Baseado no *fitness*, cada anticorpo gera novos herdeiros, chamados

de *clones* e que possuem as mesmas características do anticorpo-pai. Os clones de um anticorpo são gerados em uma medida inversamente proporcional ao seu *fitness*, ou seja, quanto menor o *fitness* de um anticorpo, mais herdeiros ele conterà. No modelo, também há o conceito de *rank*, que é calculado a partir do *fitness* e serve para classificar os anticorpos além de ser utilizado para o cálculo do número de *clones* e para a *maturação* (conceitos que serão explicados a seguir). Um clone é um anticorpo que possui as mesmas características do anticorpo-pai que o gerou. A quantidade de clones que são gerados depende da medida de afinidade (*fitness*) do anticorpo-pai. Quando o anticorpo gera n clones, eles são submetidos a um processo denominado *maturação* que utiliza alguns mecanismos com o objetivo de alterar esses clones e melhorar o desempenho (*fitness*) de cada um deles.

Após o processo de maturação dos clones, é feita uma comparação entre eles, com o intuito de determinar o melhor clone (aquele que possui o melhor *fitness*). O melhor clone é então comparado com o anticorpo-pai (anticorpo que o gerou), aquele que possuir o melhor *fitness* permanece na população, caracterizando assim o conceito de elitismo deste modelo. Esse mecanismo é o que caracteriza o princípio da seleção clonal, ele não apresenta comportamento retrógrado durante as iterações e privilegia o clone com maior grau de afinidade (DE CASTRO, L. N., VON ZUBEN, F. J., 1999), ou seja, privilegia os anticorpos mais adaptados ao ambiente (ao problema em questão). Caso o clone substitua o anticorpo, ele passa automaticamente a ser chamado de anticorpo no contexto do modelo. Essa abordagem é dita elitista (MICHALEWICZ, Z., 1996), o que assegura que as melhores soluções calculadas em cada iteração não serão perdidas. Na seção de Metodologia, mais detalhes sobre este modelo e sobre como ele foi implementado no iMaRag serão abordados.

3.3.1 Critério de dominância

Dominância é um critério utilizado para determinar se uma solução (anticorpo) é melhor que outra. Esse critério depende do problema a ser resolvido, se é um problema de minimização ou se é um problema de maximização. Dominância é um critério muito utilizado em algoritmos evolucionários que são utilizados para resolução de problemas de otimização multi-objetivos (ZITZLER, E., THIELE, L., 1999), porém também pode ser empregado para problemas mono-objetivo.

Como neste trabalho retratamos um problema de minimização mono-objetivo, dizemos que um anticorpo domina o outro se o seu *fitness* é maior que o *fitness* do outro anticorpo. De uma maneira formal podemos dizer que: Sejam **a** e **b** anticorpos, **a** domina **b** se o $fitness(\mathbf{a}) > fitness(\mathbf{b})$. Caso o $fitness(\mathbf{a}) = fitness(\mathbf{b})$ esses dois anticorpos são indiferentes entre si e no nosso caso, seriam encarados como anticorpos iguais, mesmo que seus vetores de pesos sejam diferentes.

4. Metodologia

Este capítulo apresenta o processo de desenvolvimento dos temas abordados nos capítulos 2 e 3. Vale lembrar que o trabalho propõe a implementação de um algoritmo baseado em um SIA para o treinamento de uma RNA do tipo MLP. Aqui também é descrito como essa implementação foi realizada e qual foi a abordagem utilizada para relacionar essas duas técnicas. Os algoritmos foram escritos na linguagem de programação *C#* e no ambiente de desenvolvimento *Microsoft Visual Studio 2008*.

Na seção 4.1 é feita uma descrição das bases de dados utilizadas para o trabalho, logo depois na seção 4.2 é mostrado como foi feito o tratamento dos dados. A seção 4.3 ilustra o algoritmo proposto, com detalhes de suas características e principais mecanismos.

4.1 Bases de dados

As simulações para os tipos de problemas de classificação foram realizadas com bases de dados muito conhecidas e utilizadas na literatura para esse tipo de problema: base de íris e base de vinhos. Basicamente, em sua estrutura, cada base de dados possui um número de exemplos e uma quantidade de parâmetros de entradas e saídas que é comum para todos os exemplos da base.

4.1.1 Base Íris

A base de dados Íris é uma base que possui 150 exemplos ao total de três classes de plantas (50 exemplos para cada classe): Íris-setosa, Íris-versicolor e Íris-virginica. Cada exemplo possui quatro características (atributos) que determinam a qual classe ele pertence, são elas: largura da sépala (cm), comprimento da sépala (cm), largura da pétala (cm) e comprimento da pétala (cm). Estes atributos funcionam como parâmetros de entrada e a saída seria uma entre as três classes da planta.

4.1.2 Base de Vinhos

A base de dados de Vinhos é uma base que possui três classes distintas de vinhos, chamadas somente por: Vinho 1, Vinho 2 e Vinho 3. Cada classe possui 59, 71 e 48 exemplos, totalizando 178 exemplos. Ela possui 13 atributos que caracterizam cada tipo de vinho.

4.1.3 Base de Curuá-Una

A base de dados utilizada para o problema de regressão foi a base de dados da Usina Hidrelétrica Curuá-Una. A barragem de Curuá-Una situa-se a 70 Km da cidade de Santarém, no rio Curuá-Una, afluente da margem direita do rio Amazonas. A barragem dista cerca de 850 Km da cidade de Belém. Esta base é composta por vazões naturais médias diárias afluentes ao reservatório de Curuá-Una desde 01 de janeiro de 1978 até 31 de dezembro de 2007.

4.2 Tratamento dos dados

Os dados são extraídos da base e passam por um procedimento de *normalização*. A normalização tem o objetivo de igualar a importância das variáveis que estiverem em intervalos diferentes. Quando os dados não são divididos adequadamente e possuem intervalos distintos de variação para os subconjuntos, a modelagem ficará prejudicada uma vez que as redes neurais serão inábeis para generalizar quando utilizadas em um conjunto com intervalo de variação diferente ao do treinamento, ou seja, quando submetida a um conjunto de verificação para o qual os dados não são estatisticamente compatíveis com os de treinamento (VALENÇA, Mêuser, 2009). As bases também precisam ser normalizadas em valores que não ultrapassem os limites da função de ativação utilizada para os neurônios. A função de ativação utilizada no algoritmo da rede MLP implementada para este trabalho foi a sigmóide logística (descrição na Tabela 1), cujos valores estão compreendidos entre 0 e 1, e os dados são normalizados no intervalo de 0,10 a 0,90. A normalização dos dados também garante uma maior eficiência para o *backpropagation*, que não se comporta muito bem com valores que se aproxima do extremo da função de ativação (0 e 1). Existem diversas técnicas de normalização. A

técnica utilizada no trabalho foi a conhecida como *transformação linear*, cuja fórmula é:

$$v = (z - a) \times \left(\frac{x_i - x_{\text{mín}}}{x_{\text{máx}} - x_{\text{mín}}} \right) + a \quad (4.01)$$

Onde: v é o valor normalizado; z e a são os limites máximo e mínimo para a normalização, no trabalho são 0,90 e 0,10 respectivamente; x_i é o valor do dado original; $x_{\text{mín}}$ e $x_{\text{máx}}$ correspondem ao valor mínimo e máximo para a variável em questão.

Após a etapa de normalização, o conjunto de dados é dividido em três subconjuntos independentes e aleatórios. Os valores para divisão do conjunto em três porções podem ser mudados na interface gráfica da aplicação, porém, para efeito de testes sempre se utilizou os mesmos valores de divisão: 50% da amostra é utilizada para o treinamento, ou seja, para os ajustes dos pesos da rede neural; 25% da amostra é destinada a Validação Cruzada e os 25% restantes são utilizados para efetuar os testes de desempenho da rede.

Efetuada o tratamento dos dados, a rede pode ser utilizada para o processo de treinamento e posterior processo de testes.

4.3 Processamento da rede MLP com o *backpropagation*

Após a etapa de pré-processamento dos dados, basta realizar a configuração da arquitetura da rede MLP e dos parâmetros do *backpropagation* antes de iniciar o treinamento da rede. A Figura 14 foi retirada de uma das telas de interface gráfica do sistema construído para este trabalho e apresenta um exemplo de configuração desses parâmetros.

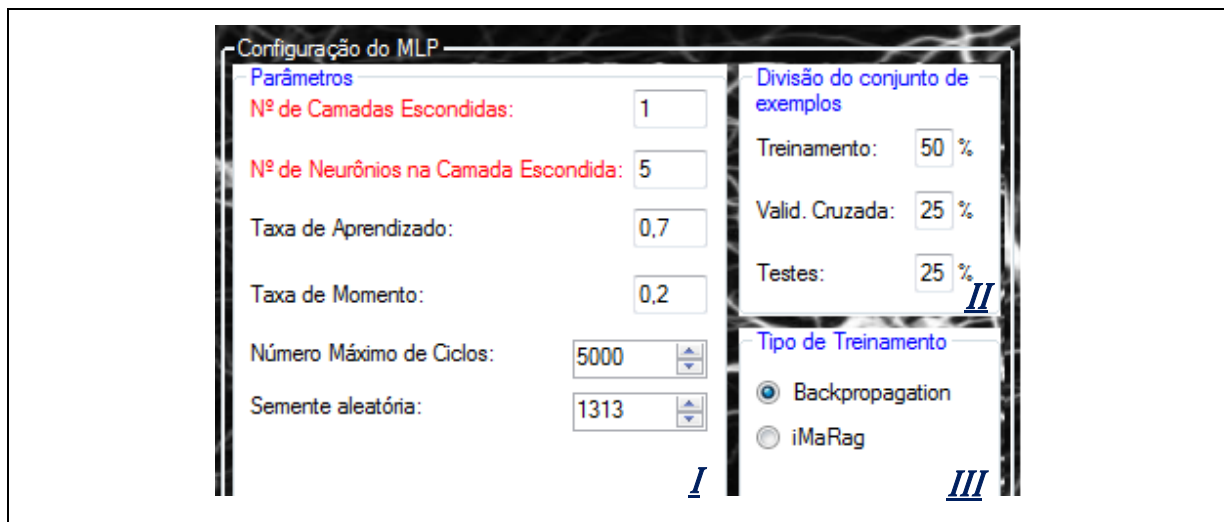


Figura 14. Configuração da arquitetura da rede e do *backpropagation*

A configuração da arquitetura da rede resume-se aos dois **parâmetros** destacados em cor vermelha na Figura 14, na porção destacada pelo símbolo 'I'. São os parâmetros de *número de camadas escondidas* e *número de neurônios em cada camada escondida*. A técnica MLP e a sua arquitetura foram discutidas na seção 2.4.

Para o *backpropagation* os quatro parâmetros (em cor preta) retratados na porção 'I' da Figura 14 (*Taxa de aprendizado*, *taxa de momento*, *número máximo de ciclos* e *semente aleatória*) precisam ser preenchidos. A taxa de aprendizado revela o quão grande é o tamanho do passo na direção da correção do erro. Portanto, quanto menor for o valor da taxa de aprendizagem, menor serão as variações dos pesos sinápticos da rede. Por outro lado, quanto maior a taxa de aprendizagem as variações nos pesos sinápticos serão maiores e podem também causar certa instabilidade na rede. A taxa de momento é um valor que é utilizado para um método simples que objetiva aumentar a taxa de aprendizagem evitando o perigo da instabilidade na rede (HAYKIN, Simon, 2001). Os dois parâmetros restantes serão explicados a seguir, inseridos no contexto do processamento geral do algoritmo.

Após as definições explicadas acima, a simulação pode ser iniciada e o primeiro processamento é no intuito de gerar os pesos iniciais da rede. Para isso, utiliza-se um gerador de números aleatórios da própria ferramenta de implementação que recebe uma semente como parâmetro, que é exatamente a semente aleatória, na porção 'I' da Figura 14. Os valores gerados estão no intervalo

[-1;1]. Utilizar uma semente para geração dos pesos aleatórios é interessante, pois podemos repetir os resultados ao utilizar a mesma configuração e semente (na subseção 4.4.3, é explicado o problema encontrado com a sensibilidade da mudança de semente na geração dos pesos aleatórios). Em seguida começa a etapa de treinamento onde os pesos serão ajustados gradativamente ao longo dos ciclos.

Como explicado na seção 2.6, o critério de parada da rede é a validação cruzada (VC). A rede interrompe o treinamento caso o erro de validação cruzada (EMQVC) fique constante por 50 ciclos seguidos ou permaneça aumentando por 100 ciclos seguidos. O outro critério de parada é o parâmetro de Número máximo de ciclos, quando esse valor é superado e a rede ainda não convergiu pelo critério de VC, o algoritmo interrompe o treinamento. Após o encerramento da fase de treinamento do algoritmo, os melhores pesos encontrados são armazenados e configurados na rede para a etapa de testes.

A etapa de testes computa a quantidade de exemplos que a rede acerta e faz o cálculo do desempenho. Para os problemas de classificação, a medida de desempenho é simplesmente a taxa de acerto, calculada pela divisão da quantidade de acertos sobre a quantidade de exemplos da base de testes. Para os problemas de regressão, além desta taxa de acerto, também é computado o Erro Médio Percentual Absoluto (EMPA), calculado pela equação abaixo.

$$EMPA(\%) = \frac{1}{N_{ex}} \times \frac{1}{N_{out}} \times \sum_{i=1}^{N_{out}} \sum_{j=1}^{N_{ex}} \frac{|V_d - V_c|}{V_d} \times 100 \quad (4.02)$$

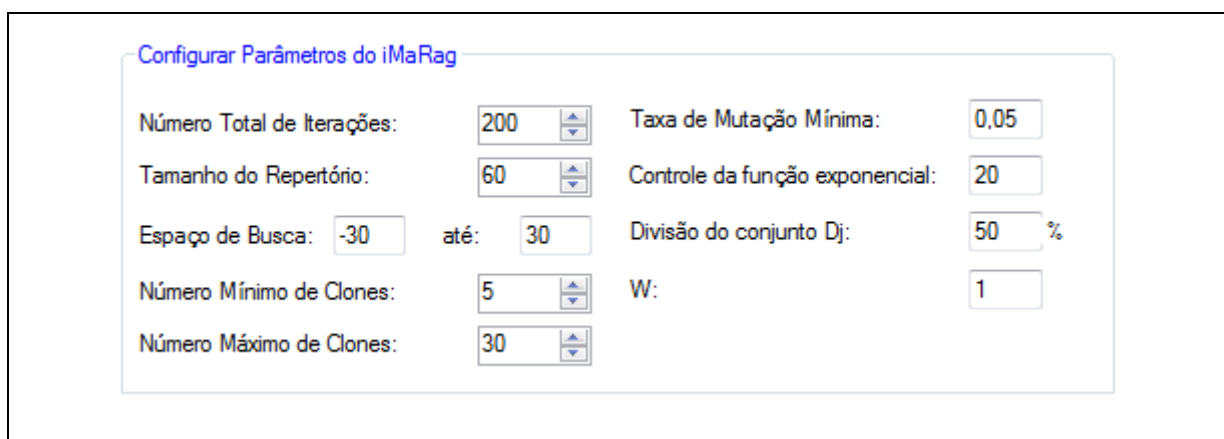
Onde: N_{ex} = Número total de exemplos da base de verificação; N_{out} = Número de saídas; V_d = Valor desejado e V_c = Valor calculado pela rede.

A porção 'II' da Figura 14 apresenta a divisão da amostra em três subconjuntos distintos, esses valores podem ser mudados pela interface, mas foram mantidos em 50% para o treinamento, 25% para VC e 25% para testes durante todas as simulações.

A porção 'III' da Figura 14 ilustra o algoritmo de treinamento que foi selecionado para o processamento da rede, neste caso, o *backpropagation*.

4.4 O algoritmo proposto: *iMaRag*

O *iMaRag* é o algoritmo baseado nos sistemas imunológicos, mais precisamente no modelo de seleção clonal, que foi implementado para este trabalho. Ele é uma simplificação do nome *Immune Artificial Algorithm* e é utilizado para o treinamento da rede MLP, substituindo o tradicional *backpropagation*. O *iMaRag*, como os demais algoritmos bio-inspirados, possui diversos parâmetros de entrada para sua configuração, e um dos objetivos ao construí-lo, foi buscar exatamente uma maneira de automatizar alguns valores desses parâmetros. Portanto, a Figura 15 é uma das telas do sistema implementado e apresenta os parâmetros para configuração do *iMaRag*. Esses parâmetros serão explicados nas próximas seções, sempre dentro do contexto do mecanismo ao qual eles pertencem no algoritmo. Alguns desses parâmetros foram embutidos com a finalidade de diminuir os parâmetros, tornando a ferramenta mais versátil e menos complicada. A seção 4.4.8 retrata a configuração final dos parâmetros para o *iMaRag*.



Configurar Parâmetros do *iMaRag*

Número Total de Iterações:	200	Taxa de Mutação Mínima:	0,05
Tamanho do Repertório:	60	Controle da função exponencial:	20
Espaço de Busca:	-30 até: 30	Divisão do conjunto D_j :	50 %
Número Mínimo de Clones:	5	W:	1
Número Máximo de Clones:	30		

Figura 15. Parâmetros de configuração (inicialmente) para o *iMaRag*

4.4.1 Representação/Mapeamento do problema

O *iMaRag*, que é baseado no modelo de seleção clonal (seção 3.3), possui um repertório \mathbf{Ab} formado por n_T anticorpos de dimensão n . Cada anticorpo \mathbf{Ab}_i codifica um vetor de valores reais, e cada valor real desse é um peso sináptico (conexão) da rede neural. Na Figura 16 é ilustrada uma arquitetura simples de rede MLP que contém apenas oito neurônios no total, sendo: dois neurônios de entrada, uma camada intermediária com quatro neurônios escondidos e dois neurônios de saída. Na figura, também podem ser vistos os valores das conexões entre os

neurônios. A Figura 16 serve para exemplificar a representação de um único anticorpo no algoritmo iMaRag.

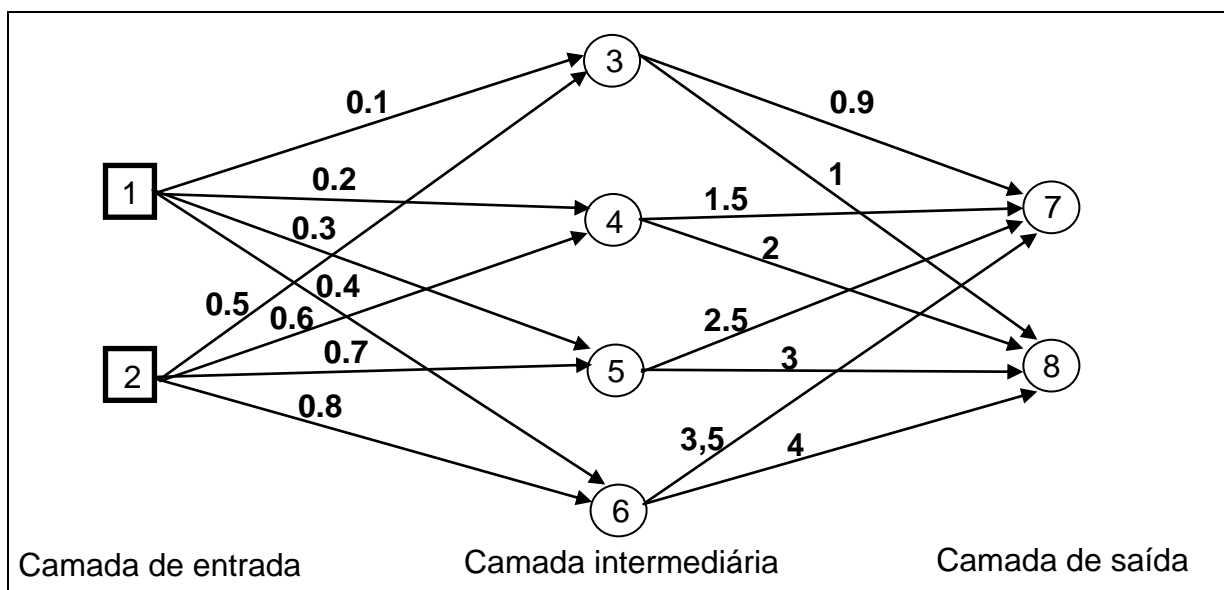


Figura 16. Arquitetura de rede MLP e seus pesos sinápticos

O vetor Ab_i que representa um anticorpo é formado pelos valores das conexões entre os neurônios da rede. Seguindo esse princípio, o anticorpo que representa essa configuração de pesos da Figura 16 seria:

$$Ab_i = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.5, 2, 2.5, 3, 3.5, 4]$$

dimensão (n) = 16 (somatório da quantidade de conexões que a rede possui).

4.4.2 Inicialização do repertório

O repertório de anticorpos é inicializado de forma aleatória. A quantidade de anticorpos para o repertório inicial é um parâmetro disponibilizado pelo usuário, através da interface gráfica do sistema e é representado pelo *Tamanho do Repertório* na Figura 15. Outro parâmetro importante para a inicialização do repertório é o *Espaço de Busca* (Figura 15). Dessa maneira, se considerarmos que os valores desses dois parâmetros são os que estão apresentados na Figura 15, sendo 60 para o tamanho do repertório e o espaço de busca no intervalo de [-30; 30] e ainda a Figura 16 como a arquitetura da rede, o algoritmo iMaRag geraria 60 anticorpos, onde cada anticorpo possuiria 16 posições e cada posição dessa teria um valor de conexão (peso) gerado aleatoriamente no intervalo de -30 a 30. A Tabela 3 exemplifica o que foi explicado nesta subseção.

Tabela 3. Exemplo de repertório de anticorpos gerado para um espaço de busca no intervalo [-30;30]

Identificação do anticorpo	Vetor com os valores gerados aleatoriamente que representa o anticorpo
1	[-15, 0.21, 2.3, 5.7, 1.3, -0.16, 8.1, 25.2, -29, 10, 1, -21, 4, -1, 17, 6]
2	[-22, 1.5, 7.1, -3.3, 0.2, 1.1, -13.9, 16, -2.4, 11, 2, -16, 8.2, 13, 27, 2]
3	[0.1, 0.04, 25, -14.1, 2.7, 1.21, -9.2, 28, -29, 5, -3, -1, -0.4, 21, 7, -8]
...	...
60	[-13, 1.3, 12.8, 7.9, 8.1, 17.7, 8.5, 29.8, -20, 11, 1, -22.3, 5, -3, -5, 3]

4.4.3 Sensibilidade à semente de geração dos pesos aleatórios

De acordo com a semente de geração dos pesos aleatórios, uma seqüência de valores, dentro de uma determinada faixa (escolhida pelo usuário, como explicado acima), é formada para utilização nos pesos que irão formar os anticorpos de um repertório inicial. Para testarmos a sensibilidade em relação à semente, a base de íris foi normalizada e embaralhada de maneira aleatória, em seguida foi fixada juntamente com uma configuração para o algoritmo e em cima disso, foram realizadas 10 simulações, variando apenas as sementes de geração aleatória. O algoritmo apresentou algumas mudanças, em relação à precisão, de acordo com a semente escolhida. A Tabela 5 mostra os resultados encontrados de acordo com as sementes escolhidas.

Tabela 4. Variação das sementes de geração de pesos aleatórios e resultados encontrados na base de Íris

Semente utilizada	Taxa de acerto
1313	100%

Semente utilizada	Taxa de acerto
56789	94,59%
87436789	94,59%
120	94,59%
56	91,89%
5656	91,89%
443399	91,89%
422489	83,78%
13	78,38%
87654	64,86%

A maioria dos resultados encontrados foram satisfatórios, mas os últimos três resultados não são razoáveis para uma base de fácil convergência como a íris e, por essa razão, um dos objetivos futuros deste trabalho é buscar uma maneira de retirar essa pequena sensibilidade que altera muito os resultados.

4.4.4 Inicialização do treinamento da rede MLP

Após a inicialização do repertório, o algoritmo entra na fase de treinamento da rede. Nesta etapa, o parâmetro *Número total de iterações* da Figura 15 é utilizado como um dos critérios de parada para o treinamento. O outro critério, assim como no *backpropagation*, é a validação cruzada. Cada anticorpo vai ter seu vetor de pesos configurado na rede e em seguida, esse conjunto de pesos é processado para toda a amostra de treinamento. Diferentemente do *backpropagation*, o conjunto de pesos do anticorpo não é ajustado a cada exemplo da amostra de treinamento. No algoritmo, o vetor de conjunto de pesos permanece inalterado e vai sendo submetido aos exemplos, e um a um o erro quadrático vai sendo computado e ao fim dos exemplos, o EMQ é calculado para então o algoritmo entrar na próxima etapa: atribuição de *fitness* e cálculo do *rank*.

4.4.5 Atribuição de *fitness* e cálculo do *rank*

Como já foi explicado na seção 3.3, o *fitness* serve para representar o quão bom o anticorpo é dentro do repertório. Ele também pode ser visto como uma medida de desempenho de um anticorpo. A medida de *fitness* para um anticorpo sempre é atribuída após o processamento deste anticorpo na base de treinamento da rede, ou seja, após o cálculo do EMQ de acordo com o vetor de pesos do anticorpo. O *fitness* de um anticorpo simplesmente é o inverso do EMQ calculado para ele, ou seja:

$$fitness = \frac{1}{EMQ} \quad (4.03)$$

Após a atribuição do *fitness*, o cálculo do *rank* é iniciado. Cada anticorpo possui um valor (inteiro) auxiliar para a atribuição do *rank*, esse valor é denominado de **SC** (soma da cardinalidade) e inicialmente é zero para todos os anticorpos. Sabendo disso, o *rank* (que também é zero inicialmente) é atribuído em dois passos:

- 1) Para cada anticorpo **Ab_i** do repertório, é feita uma comparação de dominância entre ele e todo outro anticorpo **Ab_o** do repertório. Caso **Ab_i** domine **Ab_o**, ou seja, caso o $fitness(\mathbf{Ab}_i) > fitness(\mathbf{Ab}_o)$ então o valor **SC** para o anticorpo **Ab_i** é incrementado em 1. Em outras palavras:

$$SC(\mathbf{Ab}_i) = \text{Número de elementos que } \mathbf{Ab}_i \text{ domina no repertório}$$

- 2) Para cada anticorpo **Ab_i** do repertório, é novamente feita uma comparação de dominância entre ele e todo outro anticorpo **Ab_o** do repertório. Caso **Ab_i** seja dominado por **Ab_o**, ou seja, caso o $fitness(\mathbf{Ab}_i) < fitness(\mathbf{Ab}_o)$ então o *rank* de **Ab_i** é incrementado com o **SC** do anticorpo **Ab_o**. Ou seja, o melhor anticorpo é aquele que não é dominado por ninguém, e se ele não é dominado por ninguém, não haverá nenhum incremento com valores de SC de outros anticorpos, portanto seu *rank* permanecerá igual a zero.

Matematicamente:

$$Rank(\mathbf{Ab}_i) = \sum_{\mathbf{Ab}_o \text{ domina } \mathbf{Ab}_i} SC(\mathbf{Ab}_o) \quad (4.04)$$

Após calcular o *rank* de todos os anticorpos, o repertório é ordenado de maneira crescente, ficando os melhores anticorpos (os que possuem **menor rank**) nas primeiras posições do repertório.

4.4.6 Clonagem

A clonagem é a etapa de criação de clones a partir de cada anticorpo do repertório. Como já foi explicado, um clone é um anticorpo que possui as mesmas características do anticorpo-pai que o gerou. A quantidade de clones que são gerados depende da medida de afinidade (*fitness*) do anticorpo-pai. Quanto maior o *fitness* do anticorpo-pai (ou seja, quanto menor o *rank* deste anticorpo) mais clones serão gerados. O número de clones a serem gerados para um anticorpo ($N_c(Ab_i)$) segue a fórmula a seguir:

$$N_c(Ab_i) = c_{máx} - \left\lfloor \frac{c_{máx} - c_{mín}}{Máx\{Rank(Ab)\}} \times (Rank(Ab_i)) \right\rfloor \quad (4.05)$$

Sendo: $c_{máx}$ = número máximo de clones (que é passado pelo parâmetro *Número Máximo de Clones* através da interface gráfica da ferramenta como está ilustrado na Figura 15), $c_{mín}$ = número mínimo de clones (que é passado pelo parâmetro *Número Mínimo de Clones* através da interface gráfica da ferramenta como está ilustrado na Figura 15), $Rank(Ab_i)$ = valor do *rank* do anticorpo Ab_i e $Máx\{Rank(Ab)\}$ = valor do *rank* do pior anticorpo do repertório (é o maior *rank* entre todos).

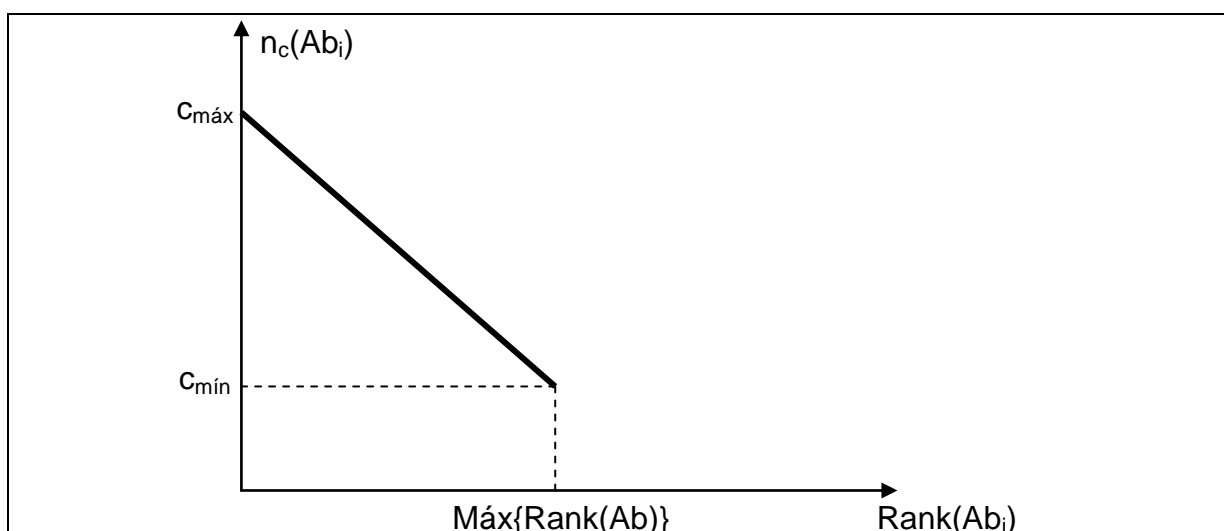


Figura 17. Número de clones gerados por anticorpo em função do $Rank(Ab_i)$

O gráfico da Figura 17 ilustra que à medida que o *rank* do anticorpo vai crescendo o número de clones que são gerados vai diminuindo. Para o anticorpo que possui *rank* = 0, o número de clones gerados será o máximo, que foi definido pelo usuário pelo parâmetro, analogamente, o pior anticorpo irá gerar o mínimo de clones também recebido como parâmetro.

4.4.7 **Maturação**

Maturação é processo em que cada clone C_j gerado é modificado por uma perturbação aleatória ($\alpha_i \cdot d_j$) com o intuito de explorar o espaço de busca e diversificar as soluções. Desta forma:

$$C_j = Ab_i + \alpha_i \cdot d_j \quad (4.06)$$

Sendo: Ab_i = anticorpo-pai que gerou o clone C_j , α_i = tamanho do passo (grau de variação) a partir da posição de Ab_i e d_j = direção do deslocamento do clone C_j .

Cálculo de α_i

A medida α_i é influenciada de acordo com o grau de afinidade do anticorpo, ou seja, quanto maior a afinidade (e por conseqüência menor o *rank* e melhor o anticorpo) menor será esse grau de variação denominado α_i . A amplitude de variação é um valor que é compartilhado entre todos os clones de um mesmo anticorpo-pai e no modelo original de seleção clonal segue a seguinte fórmula (será explicado mais adiante que houve mudanças nessa fórmula no intuito de melhorar os resultados para o nosso problema):

$$\alpha_i = \beta \times e^{\text{rank}(Ab_i)/\sigma} \quad (4.07)$$

Onde: β é a taxa de mutação, fornecido pelo usuário através da interface gráfica e representado pelo parâmetro *Taxa de Mutação Mínima* da Figura 15. σ é o valor de controle do crescimento da função exponencial, fornecido pelo usuário através da interface gráfica e representado pelo parâmetro *Controle da Função Exponencial* da Figura 15.

Cálculo de d_j

Cada clone C_j possui sua respectiva direção de perturbação d_j . De acordo com o modelo de seleção clonal, esta direção é calculada de duas maneiras:

1. Para 2/3 dos clones gerados pelo anticorpo, atribui-se ao d_j de cada clone um valor aleatório no intervalo entre -1 e 1. O objetivo aqui é a exploração do espaço de busca.
2. Para o 1/3 restante dos clones gerados, a direção d_j será calculada a partir da seguinte fórmula:

$$d_j = p_1 \times d_a + p_2 \times d_b + W \quad (4.08)$$

Onde: p_1 e p_2 são pesos aleatórios gerados entre 0 e 1; d_a refere-se ao *anticorpo* resultante da iteração imediatamente anterior. Se na iteração anterior o anticorpo foi substituído pelo clone X então d_a recebe o d_j de X, caso contrário, se o anticorpo foi melhor que o melhor clone na iteração anterior, d_a recebe o valor zero; d_b é definida como sendo a direção do clone C_j ao líder do repertório. A idéia é que o líder deve guiar a busca para regiões promissoras. Para os clones que são gerados pelo anticorpo líder atribui-se zero a d_b , para os demais d_b resulta na distância euclidiana entre o clone C_j e o líder do repertório; Por último, o W reflete um valor de equilíbrio entre a exploração local e global (PARSOPOULOS, K. E. E., VRAHATIS, M. N., 2002). Portanto, valores altos de W facilitam a exploração global (busca por novas áreas) enquanto valores pequenos facilitam a exploração local (refinamento de áreas de busca já visitadas) (BERBERT, P. C., 2008).

Diferentemente do modelo original, no algoritmo implementado a divisão para o cálculo da distância d_j pode ser modificada pelo usuário através do parâmetro *Divisão do conjunto d_j* (Figura 15). O valor para o W também é fornecido através da interface gráfica pelo parâmetro W (Figura 15). Na figura abaixo, é possível visualizar o exemplo da maturação de três clones, com o valor de $\alpha_i = 0.4$.

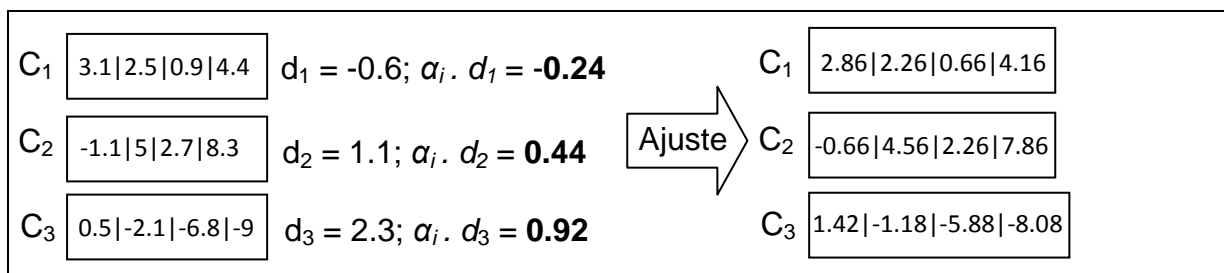


Figura 18. Processo de maturação no ajuste dos pesos dos clones

Após todos os clones de todos os anticorpos serem maturados, eles são processados na rede com o objetivo da atribuição de *fitness* de cada um e posterior comparação com o anticorpo-pai. Aquele que apresentar o melhor *fitness* é conservado no repertório e o processo se repete até que o número de iterações definido pelo usuário seja alcançado.

4.4.8 Otimização dos parâmetros e ajuste de fórmulas do modelo original para o iMaRag

Como já foi mencionado, buscamos também diminuir a quantidade de parâmetros para o iMaRag, e portanto, alguns ajustes foram feitos para se alcançar este objetivo.

Inicialmente, a equação 4.07 precisou sofrer algumas modificações para que representasse melhor a realidade do nosso problema. Como o *Controle da Função Exponencial* é um valor constante e o *rank* é um valor que usualmente começa em 0 e vai aumentando consideravelmente, o que torna o fator da exponencial $e^{\text{rank}(Ab_i)/\sigma}$ (equação 4.07) resultar em um valor muito alto. Por essa razão, algumas tentativas foram realizadas até que fosse encontrada uma maneira que tornasse os resultados mais satisfatórios e que o parâmetro de *Controle da Função Exponencial* pudesse ser embutido. Para isso, vários testes foram realizados, com três bases de dados diferentes (ver seção 4.1). A equação final que mostrou os melhores resultados ficou da seguinte maneira:

$$\alpha_i = \beta \times e^{\text{posição}(Ab_i)/20} \quad (4.09)$$

Onde: *posição*(Ab_i) retrata a posição, a colocação do anticorpo diante dos outros no repertório. O melhor anticorpo se encontra na primeira posição e o pior anticorpo na posição: (*tamanho do repertório*) – 1. Com essa abordagem, gera-se uma série exponencial que pode ser visualizada na Figura 19.

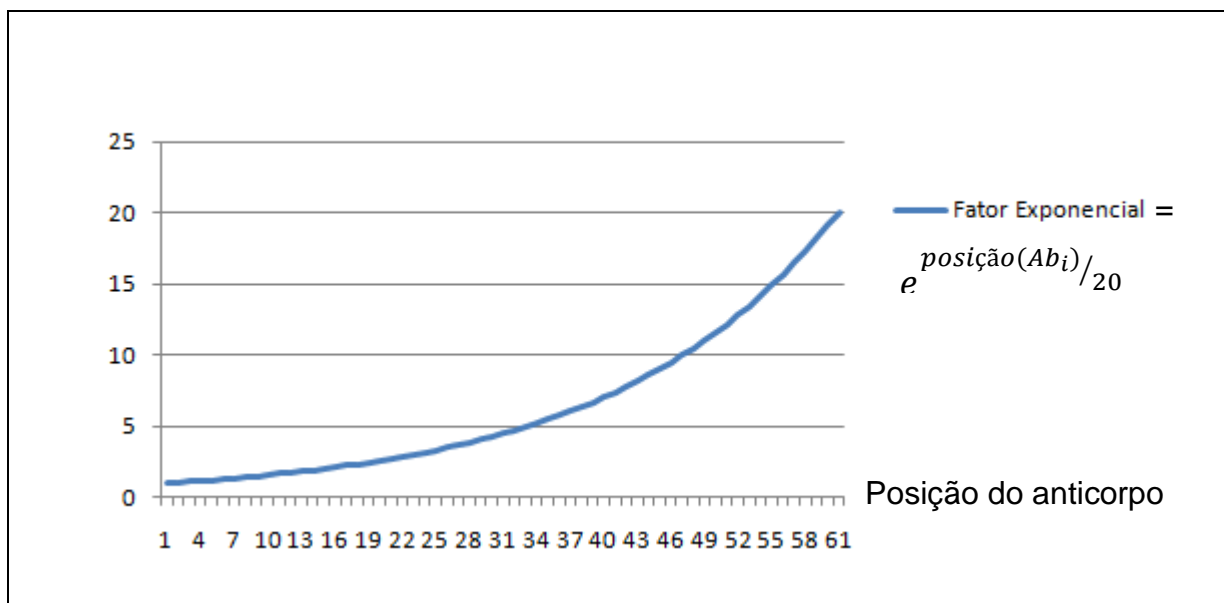


Figura 19. Curva gerada para o fator exponencial utilizado no cálculo de α_i

A fórmula 4.08 também sofreu algumas alterações. Optou-se pela supressão do valor de equilíbrio W (equação 4.10), porém o desempenho não foi satisfatório.

$$d_j = p_1 \times d_a + p_2 \times d_b \quad (4.10)$$

Então, resolvemos manter o W , porém, na tentativa de embutir seu valor e então vários valores foram testados dentro de um intervalo de $[0.1; 20]$, o valor máximo foi 20 pela natureza do problema a ser resolvido. Esse valor 20, por exemplo, caracteriza uma mudança muito drástica nos valores dos pesos dos anticorpos o que também não resulta em bom desempenho. Não foi encontrado um valor ótimo para esse parâmetro, porém uma boa faixa de valores percebida foi entre 0.5 e 2. Os resultados para valores abaixo e acima desses não se mostraram aceitáveis. Dessa maneira a equação para o cálculo do d_j permaneceu sendo a equação 4.08.

Outras modificações foram feitas, sendo no ajuste dos pesos dos clones, após o cálculo do α_i e do d_j . No modelo original, o ajuste do conjunto de pesos pertencente a um clone é feito somando o valor de cada posição do vetor ao valor resultante da multiplicação de α_i por d_j (equação 4.06). Baseando-se nisso, a maneira de fazer o ajuste dos pesos foi adaptada para a seguinte equação:

$$C_j = Ab_i + (\alpha_i \cdot d_j \times Ab_i) \quad (4.11)$$

Essa equação indica que é feita uma variação através da soma (ou subtração, depende do valor da multiplicação de α_i por d_j) de um percentual do próprio valor do peso. Por exemplo, imagine que um peso possua valor 2 e que o resultado de $\alpha_i \cdot d_j$ seja 0.3. O ajuste do peso para o clone seria $2 + 0.3 \cdot 2$ o que resultaria em um peso com o novo valor de 2.6. Essa abordagem mostrou melhorias em relação à original nos testes realizados e, portanto é a equação final utilizada no iMaRag para efetuar os ajustes dos pesos dos clones.

Para finalizar esse capítulo, a Figura 20 demonstra o esquema completo do iMaRag, ressaltando que cada anticorpo (AB - desenho em vermelho) da figura é representado por um nome (esquerda), fitness (superior) e vetor de pesos (inferior).

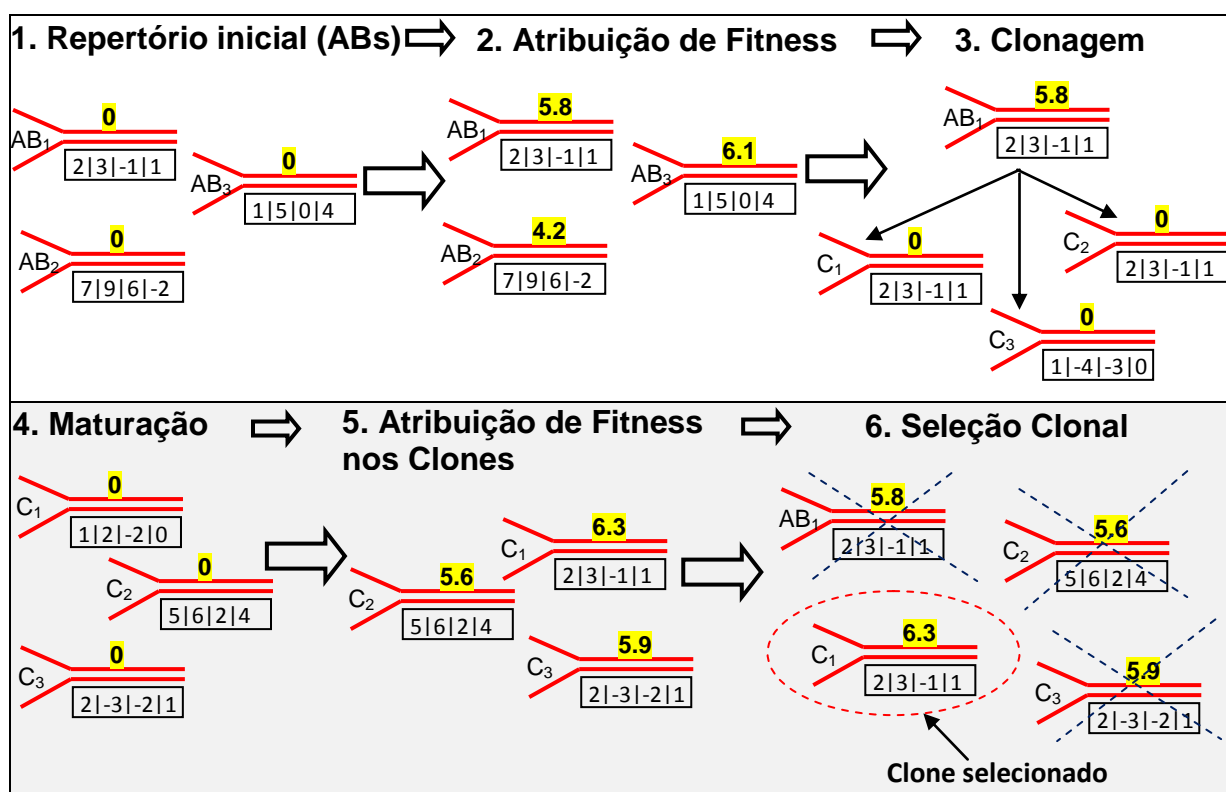


Figura 20. Esquema do iMaRag

5. Resultados

Este capítulo mostra os resultados alcançados com as técnicas discutidas no Capítulo 4. Metodologia) deste trabalho. Como já foi explicado anteriormente, o algoritmo iMaRag é comparado com o *backpropagation* em termos de eficiência e desempenho, no treinamento de uma rede do tipo MLP. As simulações, para ambos os algoritmos, foram realizadas para dois tipos de problemas: classificação e regressão.

As simulações foram realizadas sempre no mesmo computador, cuja configuração é: Processador AMD Turion X2 Dual-Core (2100 Mhz), 4GB de memória, 250GB de HD operando em um sistema operacional de 64 bits (Windows 7 Professional).

5.1 Problemas de classificação

Problemas de classificação, como o nome sugere, são problemas em que a dificuldade se encontra em identificar um objeto como sendo membro de uma classe conhecida de objetos.

5.1.1 Base Íris

Foram realizadas 30 simulações para cada algoritmo, e para manter a coerência na comparação, manteve-se a mesma arquitetura da rede (Tabela 5) a cada 10 simulações para ambos os algoritmos. Variaram-se bastante os parâmetros de cada algoritmo e os três melhores resultados obtidos, em termos de precisão (taxa de acerto) para cada arquitetura de rede (Tabela 5) e para esta base são mostrados na Tabela 6 e na 0.

Tabela 5. Configuração da rede MLP nas simulações da base de íris e vinhos

Identificação da arquitetura	Nº de camadas escondidas	Nº de neurônios escondidos
Arq-1	1	2
Arq-2	1	5
Arq-3	1	10

Na Tabela 6, os parâmetros do algoritmo ' α ' e ' β ' correspondem à *taxa de aprendizado* e à *taxa de momentum* respectivamente. O *tempo de simulação* está no formato (MM:SS) onde MM são os minutos e SS são os segundos. A taxa de acerto, como já foi dito anteriormente, é calculada pela divisão da quantidade de acertos da rede sobre a quantidade de exemplos do conjunto de testes. Nesta simulação, o conjunto de testes possui 37 exemplos. As informações sobre o *tempo de simulação* e sobre a *taxa de acerto* também valem para a Tabela 7.

Tabela 6. Resultados obtidos pela rede MLP com o algoritmo de treinamento *backpropagation*

Arquitetura da rede MLP	Parâmetros do algoritmo			Tempo de simulação	Taxa de acerto
	α	β	Nº ciclos		
Arq-1	0,1	0,5	2000	00:01	97,3%
Arq-1	0,5	0,1	3000	00:01	100%
Arq-1	0,9	0,1	5000	00:01	100%
Arq-2	0,1	0,5	2000	00:02	100%
Arq-2	0,3	0,1	3000	00:01	100%
Arq-2	0,7	0,3	5000	00:07	97,3%
Arq-3	0,1	0,1	2000	00:05	97,3%
Arq-3	0,7	0,2	3000	00:09	100%
Arq-3	0,9	0,3	5000	00:03	94,59%

Na Tabela 7, o parâmetro do algoritmo ' β ' corresponde à *taxa de mutação*. O parâmetro ' W ' é um peso de controle para busca local e global e o ' η ' é o *tamanho do repertório inicial*. Os demais parâmetros foram embutidos no algoritmo como foi explicado na seção 4.4.8. Os valores dos outros parâmetros são (Figura 15): Espaço de busca = [-30;30]; Número mínimo de clones = 5; Número máximo de clones = 30; Controle da função exponencial = 20 e Divisão do conjunto D_j = 60%.

Tabela 7. Resultados obtidos pela rede MLP com o algoritmo de treinamento iMaRag

Arquitetura da rede MLP	Parâmetros do algoritmo				Tempo de simulação	Taxa de acerto
	β	W	η	Nº ciclos		
Arq-1	0,05	0,5	60	200	01:22	94,59%
Arq-1	0,1	1	30	400	02:06	91,89%
Arq-1	0,2	1	60	600	02:17	91,89%
Arq-2	0,1	1,5	60	200	01:37	100%
Arq-2	0,15	0,5	60	400	00:59	94,59%
Arq-2	0,2	0,5	40	600	01:11	94,59%
Arq-3	0,05	1	30	200	01:41	91,89%
Arq-3	0,05	2	30	400	01:35	89,18%
Arq-3	0,1	1	40	600	02:03	91,89%

5.1.2 Base de Vinhos

Nesta base, também foram realizadas 30 simulações para cada algoritmo, seguindo o mesmo estilo das simulações com a base Íris. Utilizamos as arquiteturas de rede definidas na Tabela 5, e cada arquitetura de rede foi utilizada para 10 simulações. Novamente, os parâmetros foram sempre alterados e os três melhores resultados obtidos, em termos de precisão (taxa de acerto) para cada arquitetura de rede (Tabela 5) e para esta base são mostrados na Tabela 8 e na Tabela 9. Nesta simulação, o conjunto de testes possui 45 exemplos.

Tabela 8. Resultados obtidos pela rede MLP com o algoritmo de treinamento *backpropagation*

Arquitetura da rede MLP	Parâmetros do algoritmo			Tempo de simulação	Taxa de acerto
	α	β	Nº ciclos		
Arq-1	0,1	0,05	2000	00:11	100%
Arq-1	0,1	0,1	3000	00:01	100%
Arq-1	0,4	0,1	5000	00:01	100%
Arq-2	0,1	0,1	2000	00:02	100%
Arq-2	0,2	0,5	3000	00:01	100%
Arq-2	0,5	0,1	5000	00:02	100%
Arq-3	0,1	0,1	2000	00:02	100%
Arq-3	0,6	0,1	3000	00:12	97,78%
Arq-3	0,7	0,2	5000	00:51	97,78%

Tabela 9. Resultados obtidos pela rede MLP com o algoritmo de treinamento *iMaRag*

Arquitetura da rede MLP	Parâmetros do algoritmo				Tempo de simulação	Taxa de acerto
	β	W	η	Nº ciclos		
Arq-1	0,1	1,5	40	200	00:35	88,88%
Arq-1	0,15	2	40	400	01:13	88,88%
Arq-1	0,2	1	40	600	01:41	86,66%
Arq-2	0,1	1,5	40	200	01:05	88,88%
Arq-2	0,2	1,2	50	400	01:39	88,88%
Arq-2	0,2	2	60	600	02:28	91,11%
Arq-3	0,05	1	30	200	01:41	84,44%
Arq-3	0,1	1	30	400	01:35	84,44%
Arq-3	0,4	2	40	600	02:03	77,77%

5.2 Problemas de Regressão

São problemas em que as classes são contínuas e o objetivo é prever valores numéricos para a(s) saída(s).

5.2.1 Base de Curuá-Una

Para esta base, iremos adotar um tipo de previsão 14-12. Isto significa que cada exemplo da base consiste de 14 vazões diárias de entrada para a rede (de 1 a 13 de janeiro de 1999, por exemplo) e a sua saída são outras 12 vazões consecutivas a elas (de 14 a 25 de janeiro de 1999 seguindo o exemplo). Organizando a base desta maneira, obtém-se ao total 10932 exemplos.

Nesse contexto, foram realizadas 15 simulações para cada algoritmo. Utilizamos as arquiteturas de rede definidas na Tabela 10 Tabela 5. Como esse é um problema real, com uma base muito grande, o especialista Professor Dr. Mêuser Valença forneceu valiosas dicas para a arquitetura da rede. Novamente, os parâmetros foram sempre alterados e desta vez os cinco melhores resultados obtidos, em termos do Erro Médio Percentual Absoluto (Equação 4.02) estão ilustrados na Tabela 11 e na Tabela 12. Vale ressaltar que o EMPA menor corresponde a um melhor resultado. Nesta simulação, o conjunto de testes possui 2733 exemplos. Na tab12, o *tempo de simulação* está no formato (HH:MM:SS) onde HH são as horas, MM são os minutos e SS são os segundos que a simulação durou.

Tabela 10. Configuração da arquitetura da rede MLP para a base de Curuá-Una

Identificação da arquitetura	Nº de camadas escondidas	Nº de neurônios escondidos
Arq-1	1	12
Arq-2	1	14
Arq-3	1	18

Tabela 11. Resultados obtidos pela rede MLP com o algoritmo de treinamento
backpropagation

Arquitetura da rede MLP	Parâmetros do algoritmo			Tempo de simulação	EMPA
	α	β	Nº ciclos		
Arq-2	0,01	0,1	5000	09:41	13,13%
Arq-1	0,05	0,1	5000	08:37	14,12%
Arq-3	0,01	0,1	5000	00:01	14,45%

Tabela 12. Resultados obtidos pela rede MLP com o algoritmo de treinamento
iMaRag

Arquitetura da rede MLP	Parâmetros do algoritmo				Tempo de simulação	EMPA
	β	W	η	Nº ciclos		
Arq-2	0,2	2	60	300	06:51:27	35,14%
Arq-1	0,2	1	60	300	07:26:45	46,91%
Arq-3	0,2	1,5	60	300	09:16:10	48,02%

6. Conclusão

Este trabalho constituiu uma iniciativa pioneira no que diz respeito à utilização de um algoritmo inspirado em SIAs para o treinamento de uma rede MLP. Foi utilizado um modelo do SIA, o de *seleção clonal*, para ser a base na construção do algoritmo iMaRag (Immune Artificial Algorithm) e várias modificações foram realizadas para adaptá-lo ao problema de treinamento da rede.

Os resultados encontrados para as três bases foram satisfatórios, contudo não superaram em termos de desempenho, o algoritmo tradicional (*backpropagation*) utilizado para treinamento de redes MLP. Em relação à precisão, os resultados foram semelhantes aos encontrados com o *backpropagation*.

A principal vantagem do iMaRag em relação ao *backpropagation* é que, em sua versão final, os parâmetros foram embutidos tornando-o mais simples e de mais fácil manuseamento por um usuário. O iMaRag também foi projetado para suporte ao paradigma multi-objetivo, por essa razão, é possível que ele possa alcançar melhores resultados do que ele é capaz atualmente, principalmente em relação a problemas mais complexos.

6.1 Objetivos Futuros

O primeiro objetivo é buscar uma melhoria no desempenho do algoritmo iMaRag principalmente quando utilizado em problemas simples que possibilitem uma rápida convergência. Outros objetivos de extensão deste trabalho são:

- Realização de mais simulações com o intuito de encontrar uma configuração adequada a maior parte dos problemas;
- Tornar a rede construtiva, ou seja, estabelecer um método que permita a obtenção da configuração da arquitetura da rede neural de forma automática.
- Buscar uma maneira de tornar o algoritmo menos sensível a semente inicial de geração dos pesos aleatórios;

- Tornar o algoritmo multi-objetivo para realizar o treinamento visando minimizar funções como Erro Médio Quadrático, Erro Médio Percentual Absoluto, Índice de Concordância, Índice de erro Relativo de Pico, entre outros.

Referências

AZEVEDO, Frederico A.C., CARVALHO, Ludmila R.B., GRINBERG, Lea T., FARFEL, José Marcelo, FERRETTI, Renata E.L., LEITE, Renata E.P., FILHO, Wilson Jacob, LENT, Roberto e HERCULANO-HOUZEL, Suzana. **Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain.** The Journal of Comparative Neurology, v. 513, p. 532-541, 2009.

BECKER, S. **Unsupervised learning procedures for neural networks.** International Journal of Neural Systems, v. 2, p. 17-33, 1991.

BERBERT, Priscila Cristina. **Sistema Imunológico Artificial para Otimização Multiobjetivo.** Dissertação de Mestrado, Universidade Estadual de Campinas, Campinas, SP, Brasil, 2008.

BRAGA, Antônio, CARVALHO, André, LUDEMIR, Teresa. **Redes Neurais Artificiais: Teoria e Aplicações.** 1ª edição. Rio de Janeiro: LTC, 2000. 262 páginas.

CARVALHO, Marcio Ribeiro de. **Uma Análise da Otimização de Redes Neurais MLP por Enxames.** Dissertação de Mestrado, Universidade Federal de Pernambuco, Recife, PE, 2007.

CLARINDASOUSA, Imagem original disponível em: <<http://clarindasousa.no.sapo.pt/images/sinapse.jpg>> Acesso em: 25 de março de 2010.

CYBENKO, George. **Continuous valued neural networks with two hidden layers are sufficient.** Technical report. Department of Computer Science, Tufts University, 1988.

CYBENKO, George. **Approximation by superposition of a sigmoid function.** Mathematics of Control, Signals and Systems, p. 303-314, 1989.

DASGUPTA, Dipankar. **Advances in Artificial Immune Systems,** In: IEEE Computational Intelligence Magazine, v. 1, p. 40-49, 2006.

DASGUPTA, Dipankar. **Artificial Immune Systems and Their Applications,** 1ª edição. Berlim: Springer, 1998. 300 páginas.

DE CASTRO, L. N. **Engenharia Imunológica: Desenvolvimento e Aplicação de Ferramentas Computacionais Inspiradas em Sistemas Imunológicos Artificiais**. Tese de Doutorado, Universidade Estadual de Campinas, Campinas, SP, Brasil, 2001.

http://www.dca.fee.unicamp.br/~vonzuben/research/lnunes_dout/index.html.

DE CASTRO, L. N. E VON ZUBEN, F. J. **Artificial Immune Systems: Part I – Basic Theory and Applications**. Technical Report – RT DCA 01/99, p. 95, 1999.

FAHLMAN, Scott. **An empirical study of learning speed in backpropagation networks**. Technical report, Carnegie Mellow University, 1988.

HAGAN, Martin e MENHAJ, Mohammad. **Training feedforward networks with the Marquardt algorithm**. IEEE Transactions on Neural Networks, 5(6): 198-993, 1994.

HAYKIN, Simon. **Redes Neurais: Princípios e Prática**. 2ª edição. Porto Alegre: Bookman, 2001. 900 páginas.

KLEIN, J. **Immunology**. Blackwell Scientific Publications, 1990.

KOHONEN, Teuvo. **Self-organized formatation of topologically correct feature maps**. Biological Cybernetics v. 43, p. 59 – 69, 1982.

KULKARNI, Arun D. **Computer Vision and Fuzzy-Neural Systems**. New Jersey. Prentice Hall PTR, 2001.

MCCULLOCH, W. S. e PITTS, W. **A logical cauculus of the ideas immanent in nervous activity**. Bulletin of Mathematical Biophysics, p. 115 – 133, 1943.

MENDEL, J. M. e MCLAREN, R. W. **Reinforcement-learning control and pattern recognition systems**. Adaptative, Learning and Pattern Recognition Systems: Theory and Applications v. 66, p. 287-318, 1970.

MICHALEWICZ, Z. **Genetic Algorithms + Data Structures = Evolution Programs**. 3ª edição. Berlin: Springer-Verlag and Heidelberg GmbH & Co. K, 1996. 387 páginas.

MICHELI-TZANAKOU, E. **Computational Intelligence: A Tutorial**. In: Proceedings of the IEEE International Symposium, 1997, Guimarães, Portugal, p. 15-20.

NIELSEN, R. **Theory of the Backpropagation Neural Network**. International Joint Conference on Neural Networks. New York: IEEE TAB Neural Network Committee, 1989.

OSÓRIO, Fernando. **Redes Neurais Artificiais: Do Aprendizado ao Aprendizado Artificial**. I Fórum de Inteligência Artificial, 1999, Canoas, BR.

PARSOPOULOS, K. E. E VRAHATIS, M. N. **Particle swarm optimization method in multiobjective problems**. In: Proceedings of the ACM Symposium on Applied Computing, 2002, Madrid, Spain, p. 11-14.

QIANG Gao, KEYU Qi, YAGUO Lei, ZHENGJIA He. **An Improved Genetic Algorithm and Its Application in Artificial Neural Network Training**. In: Information, Communications and Signal Processing, Fifth International Conference on, 2005, p. 357-360.

RIEDMILLER, Martin. **Rprop – description and implementation details**. Technical report, University of Karlsruhe, 1994.

ROSENBLATT, Frank. **The perceptron: probabilistic model for information storage and organization in the brain**. Psychological Review, p. 386 – 408, 1958.

SCIENCEBLOGS, Imagem original disponível em: <http://scienceblogs.com/purepedantry/upload/2006/07/neuron.JPG> Acesso em: 25 de março de 2010.

SLOWIK, A., BIALKO, M. **Training of artificial neural networks using differential evolution algorithm**. In: Human System Interactions Conference on, 2008, p. 60-65.

STONE, M. **Cross-Validatory choice and assessment of statistical predictions**. Journal of the Royal Statistical Society, v. B36, p. 111-136, 1974.

STONE, M. **Cross-Validation: A review**. Mathematische Operationsforschung Statistischen, Serie Statistics, v. 9, p. 127-139, 1978.

VALENÇA, Mêuser. **Fundamentos das Redes Neurais: Exemplos em Java.**
2ª edição. Recife: Livro Rápido, 2009. 384 páginas.

ZITZLER, E., THIELE, L.; **Multiobjective evolutionary algorithms – A comparative case study and the strength Pareto approach.** Transactions on Evolutionary Computation, v. 3, p. 257-271, 1999.