ESCOLA
POLITÉCNICA DE
PERNAMBUCO

Um Sistema Híbrido Baseado em Rede Neural e Colônia de Formigas

Trabalho de Conclusão de Curso

Engenharia da Computação

Aluno: Saulo Medeiros de Oliveira Corrêa dos Santos

Orientador: Prof. Dr. Mêuser Jorge Silva Valença

UNIVERSIDADE
DE PERNAMBUCO

ESCOLA
POLITÉCNICA DE
PERNAMBUCO

Saulo Medeiros de Oliveira Corrêa dos Santos

Um Sistema Híbrido Baseado em Rede Neural e Colônia de Formigas

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.

Dedico este trabalho aos meus pais, Natália e Luiz Alfredo e a minha namorada Lorena, que tornaram possível minha conquista acadêmica.

Agradecimentos

Agradeço aos meus pais, Natália Medeiros de Oliveira Corrêa dos Santos e Luiz Alfredo Albuquerque Corrêa dos Santos por terem proporcionado toda educação e apoio necessários para que este objetivo fosse realizado.

À minha namorada, Lorena Graciely Neves Tablada, que me auxiliou na passagem de todas as cadeiras deste curso e por sempre estar presente.

Agradeço ao meu orientador, Prof. Dr. Mêuser Jorge Silva Valença, por mostrar o caminho para realização deste projeto e por toda a ajuda durante o seu desenvolvimento.

Resumo

Redes Neurais Artificiais têm sido cada vez mais adotadas para resolução de problemas de classificação e previsão. Sua aplicação justifica-se pela obtenção de resultados mais precisos em comparação aos obtidos com outras técnicas estatísticas tradicionais. Para que a sua utilização seja generalizada, ou seja, resolva problemas linearmente e não linearmente separáveis é necessário que se utilize uma Rede Neural do tipo Multilayer Perceptron, a qual faz uso de um algoritmo de aprendizagem conhecido como *Backpropagation*. Este algoritmo tem alguns pontos negativos no seu desempenho como: convergência muito lenta e possibilidade de ficar preso facilmente em mínimos locais. Este trabalho tem como objetivo utilizar um algoritmo baseado em colônia de formigas para realizar o treinamento da rede neural. Com isso espera-se melhorar o desempenho de sistemas que fazem uso de redes neurais.

Abstract

Artificial Neural Networks have been increasingly adopted to resolve problems of classification and prediction. Its application is justified by obtaining more accurate results than those obtained with other traditional statistical techniques. For its use to be widespread, ie, solve not linearly and linearly separable problems it is necessary to use a Neural Network Multilayer Perceptron, which uses a learning algorithm called Backpropagation. This algorithm has some drawbacks in its performance such as very slow convergence and a possibility to easily get stuck in local minima. This study proposes to use an algorithm based on an ant colony to do the neural network training. Thus it is expected to improve the performance systems that make use of neural networks.

Sumário

Resumo v

Abstract vi

Sumário vii

Índice de Figuras ix

Índice de Tabelas x

Tabela de Símbolos e Siglas xi

Capítulo 1 Introdução 12

1.1 Motivação e Problema 12

1.2 Objetivos 13

1.2.1 Objetivo Geral 13

1.2.2 Objetivos Específicos 13

1.3 Estrutura da Monografia 14

Capítulo 2 Revisão Bibliográfica 15

2.1 Redes Neurais Artificiais 15

2.1.1 Neurônios Biológicos 15

2.1.2 Neurônios Artificiais 16

2.1.3 *Multilayer Perceptron* (MLP) 18

2.1.4 *Backpropagation* 19

2.2 Colônia de Formigas 19

2.2.1 Comportamento da colônia 20

2.2.2 Modelo computacional da colônia 21

2.3 Sistemas Híbridos 23

Capítulo 3 Metodologia 24

3.1 Bases de Dados 24

3.2 Pré-Processamento 26

3.3 Modelo de Formiga 27

3.4 Inserindo Colônia de Formigas na MLP 28

3.5 Modificações Aplicadas 30

Capítulo 4 Testes e Resultados 33

4.1 Resultados do SH para classificação 33

4.2 Resultados do SH para previsão 36

Capítulo 5 Conclusões e Trabalhos Futuros 38

5.1 Conclusões 38

5.2 Trabalhos Futuros 39

Bibliografia 40

Índice de Figuras

Figura 1. Neurônio Biológico 16

Figura 2. Modelo matemático do neurônio artificial por McCulloch e Pitts 16

Figura 3. Rede MLP 18

Figura 4. Colônia de Formigas 20

Figura 5. Esquema de comportamento de uma formiga.....22

Figura 6. Pseudocódigo do algoritmo baseado na colônia de formigas 23

Figura 7. Arquitetura da Rede MLP utilizada.....29

Figura 8. Tela do Aplicativo desenvolvido. 34

Figura 9. Gráfico de validação cruzada para o melhor caso para classificação. 35

Figura 10. Gráfico de validação cruzada para o melhor caso de previsão. 37

Índice de Tabelas

Tabela 1. Amostra da base de dados das Flores Iris. 25

Tabela 2. Amostra da base de dados da Usina Curuá-Una. 25

Tabela 3. Base Iris após pré-processamento. 26

Tabela 4. Base Curuá-Una após pré-processamento. 27

Tabela 5. Valores dos parâmetros que obtiveram o melhor resultado de classificação. 35

Tabela 6. Valores dos parâmetros que obtiveram o melhor resultado de previsão. 36

Tabela de Símbolos e Siglas

RNAs – Redes Neurais Artificiais

MLP – Redes Neurais Artificiais Multilayer Perceptron

IA – Inteligência Artificial

SHs – Sistemas Híbridos

ONS – Operador Nacional do Sistema Elétrico

EMQ – Erro Médio Quadrático

EPMA – Erro Percentual Médio Absoluto

Capítulo 1

Introdução

Este capítulo descreve o problema e a motivação para o desenvolvimento deste projeto. Além de expor seus principais objetivos e ao final, revelar o conteúdo dos próximos capítulos.

1.1 Motivação e Problema

Redes Neurais Artificiais (RNAs) são sistemas paralelos distribuídos compostos por unidades de processamento simples (nodos) que calculam determinadas funções matemáticas (normalmente não lineares). Tais unidades são dispostas em uma ou mais camadas e interligadas por um grande número de conexões, geralmente unidirecionais. Na maioria dos modelos essas conexões são associadas a pesos, os quais armazenam o conhecimento representado no modelo e servem para ponderar a entrada recebida por cada neurônio da rede. O funcionamento destas redes é inspirado em uma estrutura física concebida pela natureza: o cérebro humano. (BRAGA, CARVALHO, & LUDERMIR, 2000)

No processo de execução das RNAs, existe a fase de aprendizagem, na qual é passado como entrada da rede um conjunto de exemplos de entrada onde é fornecido também o resultado de saída para cada exemplo. Após o calculo da saída para cada exemplo é feita uma comparação com seus respectivos resultados já conhecidos. Esta operação permite que a rede ajuste seus pesos para conseguir um resultado mais próximo do desejado.

Em uma rede de pequeno porte, por exemplo, uma rede com 4 entradas, 1 saída e poucos dados para treinamento, é factível que se faça o ajuste dos pesos através do algoritmo de aprendizado até manualmente (o que ocorre normalmente apenas em exercícios acadêmicos). Porém em uma rede de grande porte, por exemplo, 150 entradas, 50 saídas e vários exemplos, é inviável fazer manualmente os cálculos de treinamento desta rede, necessitando assim de um mecanismo automatizado.

Para realizar este processo de aprendizado é comumente utilizado o algoritmo *Backpropagation*. Este algoritmo consiste na generalização da regra delta, ou também conhecido como técnica do gradiente descendente. Sua execução se dá através de duas etapas: na primeira etapa calcula-se o sinal de saída e o erro propagando o sinal da entrada para a saída; na segunda etapa os erros são propagados da saída para a entrada ajustando os valores dos pesos de acordo com a regra delta generalizada. (VALENÇA, 2009)

Com a vasta utilização do *Backpropagation*, foram encontrados alguns pontos negativos no seu desempenho como: convergência muito lenta e possibilidade de ficar preso facilmente em mínimos locais. (CARVALHO, 2007)

Em função das limitações do algoritmo *Backpropagation*, várias técnicas de otimização para realizar o treinamento de redes neurais têm sido propostas. Dentre estas técnicas de otimização substitutas existe a técnica de Enxame de Partículas (CARVALHO, 2007), a técnica do Time Assíncrono (SAITO JUNIOR, NASCIMENTO JUNIOR, & YONEYAMA, 1999), que combina várias técnicas, a técnica da Colméia de Abelhas (PHAM, KOÇ, GHANBARZADEH, & OTRI, 2006), entre outras.

Este projeto visa integrar uma Rede Neural Artificial *Multilayer Perceptron* (MLP) a um algoritmo de aprendizado baseado em colônia de formigas do tipo *Pachycondila apicalis* (MONMARCHÉ, VENTURINI, & SLIMANE, 2000) (VINCENTE, 2006), e verificar a viabilidade de seu uso para treinamento de uma rede neural avaliando inclusive se esta rede assim treinada terá o seu desempenho melhorado.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver uma RNA do tipo MLP e um algoritmo de aprendizagem baseado no comportamento de uma colônia de formigas do tipo *Pachycondila apicalis*.

1.2.2 Objetivos Específicos

- Implementar uma Rede MLP.
- Desenvolver o algoritmo de aprendizado da rede baseado na colônia de formigas.

- Construir um sistema híbrido baseado na rede neural e no algoritmo de aprendizado por colônia de formigas.
- Realizar a comparação dos resultados deste sistema híbrido com os resultados de uma MLP utilizando como algoritmo de aprendizagem o *Backpropagation*.

1.3 Estrutura da Monografia

O Capítulo 2 agrupa todo o conhecimento teórico fundamental para o entendimento deste trabalho. Para isto define RNAs, detalha a técnica baseada na colônia de formigas *Pachycondila apicalis* e aborda o sistema híbrido baseado em Redes Neurais e colônia de formigas. O próximo capítulo revela todo o processo percorrido para alcançar os objetivos pretendidos. Em seguida, o Capítulo 4 mostra os resultados obtidos em conseqüência de simulações da aplicação desenvolvida. E por último, no Capítulo 5 são exibidas as conclusões obtidas e os trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

2.1 Redes Neurais Artificiais

Esta técnica de Inteligência Artificial (IA) foi desenvolvida inspirada no comportamento do cérebro humano. O cérebro é constituído de bilhões de neurônios (componente principal) que se interconectam, obtendo a capacidade de processar e se comunicar entre eles.

De maneira geral, pode-se definir uma Rede Neural Artificial como um sistema constituído por elementos de processamento interconectados, chamados neurônios, os quais estão dispostos em camadas (uma camada de entrada, uma ou várias camadas intermediarias e uma camada de saída) e são responsáveis pela não-linariadade e pela memória da rede. (VALENÇA, 2009)

Para compreender o funcionamento de uma RNA, se faz necessário a estrutura e o comportamento dos neurônios biológicos.

2.1.1 Neurônios Biológicos

Os neurônios são compostos por 3 partes: corpo celular, axônio e dendritos. O corpo celular é a parte principal da célula, onde se encontram o núcleo, ribossomos, reticulo endoplasmático e mitocôndria. O axônio é a projeção da célula que transporta a mensagem eletroquímica (impulso nervoso) pela extensão da célula. Os dendritos é o componente responsável pela comunicação do neurônio com outras células. A Figura 1 representa um neurônio biológico e uma conexão sináptica.

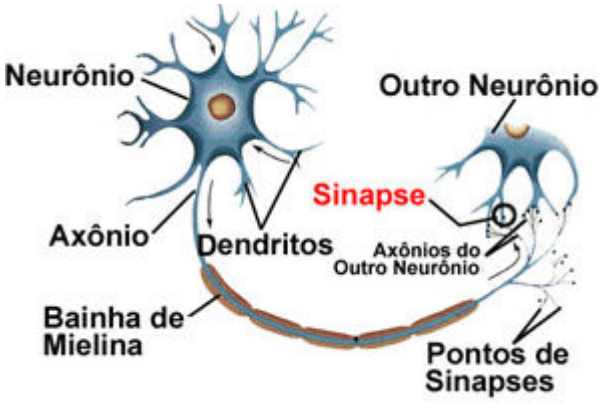

Figura 1. Neurônio Biológico

A conexão entre dois neurônios (Figura 1) ocorre entre a terminação do axônio de um com os dendritos do outro, esta região é denominada de sinapse. Pelos dendritos os neurônios recebem sinais de provenientes de outros neurônios, e os encaminham ao corpo da célula, onde será processado e um novo sinal produzido.

2.1.2 Neurônios Artificiais

O principal componente de uma rede neural artificial é o neurônio artificial. O primeiro modelo de representação matemática do neurônio biológico foi proposto por McCulloch e Pitts em 1943.

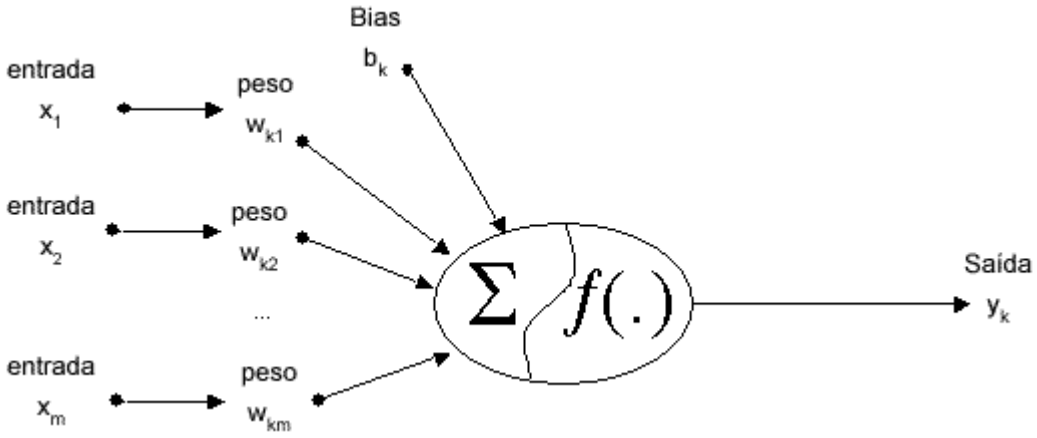

Figura 2. Modelo matemático do neurônio artificial por McCulloch e Pitts

O modelo, que representa de forma simplificada o neurônio biológico utilizando uma regra de propagação e uma função de ativação (Figura 2), contém entradas x_1, x_2... x_m (representam os dendritos) e apenas um terminal de saída y (representa o axônio). Para emular o comportamento das sinapses, os terminais de

entrada do neurônio têm pesos acoplados w_{k1}, w_{k2}... w_{km} cujos valores podem ser positivos ou negativos, dependendo de as sinapses correspondentes serem inibitórias ou excitatórias. O efeito de uma sinapse particular i no neurônio pós-sináptico é dado por $x_i w_i$. Os pesos determinam "em que grau" o neurônio deve considerar sinais de disparo que ocorrem naquela conexão (BRAGA, CARVALHO, & LUDERMIR, 2000).

A propagação de um sinal no neurônio biológico ocorre quando o somatório dos impulsos recebidos ultrapassa o seu limiar de excitação. Para o neurônio artificial ocorre a soma ponderada dos valores de entrada (net), equação 2.1, que resultará no parâmetro utilizado na função de ativação $f(net)$.

$$net_i = \sum_{j=1}^{n} w_{ij} x_j \tag{2.1}$$

A partir do modelo de McCulloch e Pitts, foram derivados outros modelos que permitem números reais como saída e diferentes funções de ativação. O primeiro modelo utiliza a função degrau, equação 2.2, como função de ativação. Esta função permite como saída o valor 0 ou o valor 1. Para uma rede MLP a função de ativação mais utilizada é a função sigmoidal logística, equação 2.3, esta função permite valores reais entre 0 e 1 como saída.

$$f(net_i) = \begin{cases} 1, \forall\ net_i \geq 0 \\ 0, \forall\ net_i < 0 \end{cases} \tag{2.2}$$

$$y_i = \frac{1}{1 + e^{-net_i}} \tag{2.3}$$

A definição da arquitetura de um RNA é de extrema importância, pois esta define o tipo de problema que pode ser processado pela rede. Uma rede neural que não possui camada escondida de neurônios (*Perceptron*, *Adaline*), somente consegue resolver problemas linearmente separáveis. Já as redes com pelo menos uma camada escondida permitem a resolução de problemas não linearmente separáveis. Este projeto busca a melhora do desempenho de uma arquitetura de rede com pelo menos uma camada escondida, ou seja, as Redes *Multilayer Perceptron* (MLP).

2.1.3 *Multilayer Perceptron* (MLP)

O modelo de RNA *Perceptron*, é uma rede simples onde existem várias unidades de processamento conectadas a uma camada de saída. Uma MLP (Figura 3) é uma generalização da rede *Perceptron* com a adição de pelos menos uma camada intermediária. Esta camada intermediária é a responsável pela não linearidade da rede, o que permite as redes MLP resolverem problemas reais.

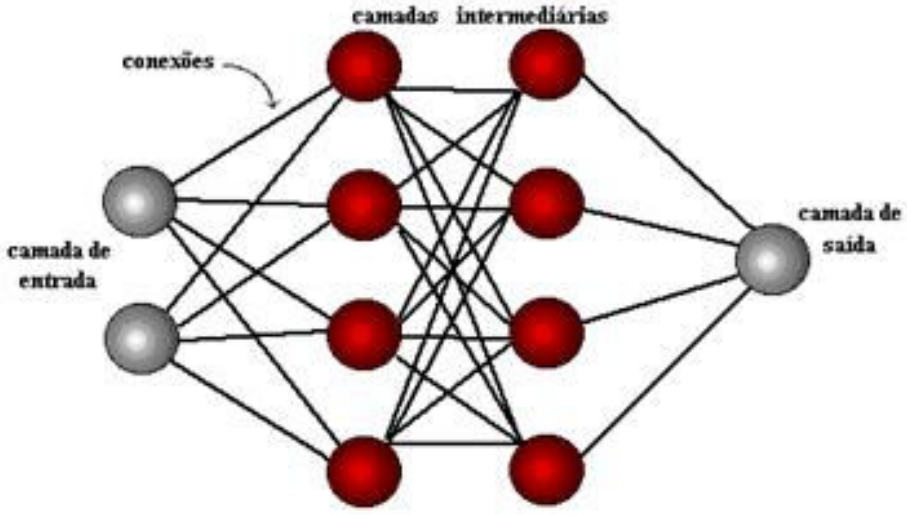

Figura 3. Rede MLP

Em uma rede multicamadas, o processamento realizado por cada neurônios é definido pela combinação dos processamentos realizados pelos neurônios da camada anterior que estão conectadas a ele (BRAGA, CARVALHO, & LUDERMIR, 2000).

Para uma rede neural conseguir bons resultados é necessário que os pesos de suas conexões sejam ajustados até que se estabeleça um conjunto de pesos ótimos em função da minimização de uma função objetivo. Este processo de minimização para estabelecimento dos pesos ótimos é chamado de treinamento da rede neural, e é realizado através da execução de um algoritmo de otimização.

O algoritmo tradicionalmente utilizado para treinamento das redes MLP consiste numa generalização da regra delta, conhecido como algoritmo *Backpropagation* (VALENÇA, 2009).

2.1.4 *Backpropagation*

As redes MLP são também redes com treinamento supervisionado de tal forma que o problema encontrado durante o treinamento destas é que, com a inclusão de pelo menos uma camada intermediaria não se conhece o erro desta camada, necessário para realizar o reajuste dos pesos. Logo, o algoritmo *Backpropagation* resolve este problema por realizar uma propagação recursiva dos erros (VALENÇA, 2009).

A aprendizagem da rede ocorre através de duas etapas: a etapa *forward* e a etapa *backward*. Na primeira etapa os sinais são propagados no sentido progressivo para definir a saída da rede. Ao final desta etapa se pode calcular o erro na camada de saída pela diferença entre a saída desejada e a calculada. Na etapa seguinte o erro é propagado recursivamente da saída até a entrada permitindo assim que seja realizado o ajuste dos pesos através da regra delta generalizada.

Um ponto importante a ser considerado durante o treinamento de uma rede neural é que o aprendizado demasiado da rede pode causar *overfitting*, ou seja, após certa quantidade de ciclos de aprendizado, a rede pode começar a memorizar os exemplos de entrada. Isto tem como conseqüência a piora da capacidade de generalização da rede. Para evitar isto durante o treinamento de uma MLP se utiliza um critério de parada. Um critério de parada bastante popular é conhecido como validação cruzada.

Para realizar a validação cruzada em uma rede é necessário reservar uma percentual, normalmente de 25%, da base de dados. Ao final de cada ciclo a rede é rodada com a base de validação onde é calculado o seu erro médio quadrático. Quando for verificado que o valor do erro da validação começou a subir, o treinamento da rede é interrompido.

2.2 Colônia de Formigas

Algoritmos baseados em comportamentos naturais são criados na tentativa de se obter um melhor desempenho em um determinado problema já calculado pelos meios mais comuns, como cálculos probabilísticos. Em (MONMARCHÉ, VENTURINI, & SLIMANE, 2000) foi desenvolvido um algoritmo baseado em uma

colônia de formigas caçadoras, encontrada no México, denominadas *Pachycondyla apicalis*.

Este algoritmo foi criado com o objetivo de aperfeiçoar o desempenho para algoritmos de busca.

2.2.1 Comportamento da colônia

As formigas estabelecem seu ninho e a partir dele elas criam áreas de caça a um raio de aproximadamente 10 metros. Estas áreas são distribuídas uniformemente ao redor do ninho com uma amplitude de aproximadamente 2,5 metros, onde as formigas exploram em busca de alimentos, ver figura 4. Desta forma as formigas conseguem cobrir uma grande superfície ao redor do ninho, formando um mosaico de pequenas regiões. Periodicamente ocorrem a mudanças de localização do ninho. Estas mudanças se devem ao fato da falta de alimento no local ou o ninho já não agrada mais a colônia. Para que ocorra a mudança de local, um complexo processo tanto de busca de um novo local como no deslocamento da colônia precisa ser executado.

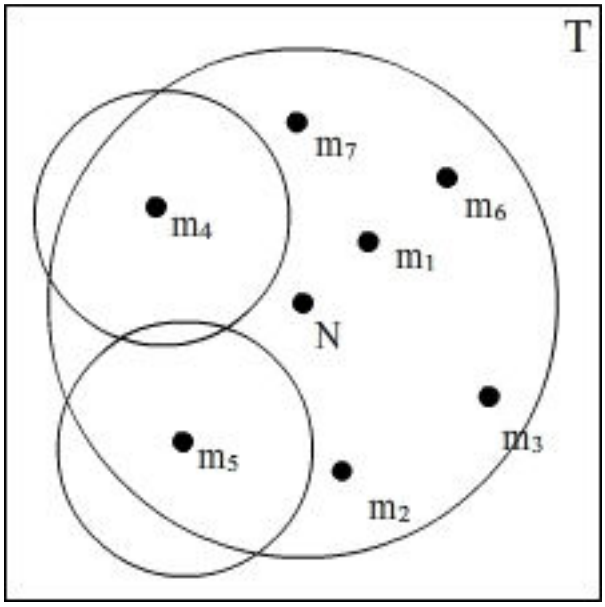

Figura 4. Colônia de Formigas

Após definir suas áreas de caça as formigas agem da seguinte forma: escolhem aleatoriamente uma de suas áreas armazenadas na memória e começam a procura por uma presa. Ao conseguirem sucesso na sua busca, esta região é memorizada no intuito de retornarem ao mesmo local em busca de mais comida.

Quando uma presa é encontrada as formigas deixam marcas no caminho até o ninho para poderem voltar na próxima exploração e recomeçar a sua caça no último ponto de sucesso. Quando a formiga demora a encontrar comida em uma determinada região, ela tende a procurar em outra região. Sempre que uma presa é capturada, esta é levada até o ninho.

2.2.2 Modelo computacional da colônia

Considerando uma população de m formigas m_1, m_2,... m_n pertencentes ao espaço de busca T, que tentam minimizar a função $f : T \rightarrow \mathbb{R}$. Cada ponto $t \in T$ é uma solução válida para o problema.

A área de busca de um ponto t é definida como círculo com centro em t e raio máximo r. Este raio define a amplitude de exploração em torno de t.

O algoritmo baseado nesta colônia de formigas pode ser descrito como a seguir. Inicialmente é estabelecido de forma aleatória a localização do ninho N. localização esta que deve pertencer a T. Então são definidas as formigas e suas regiões de busca. Com as áreas de busca definidas as formigas passam a explorar as mesmas. A cada B busca das formigas, o seu resultado é verificado em $f(B)$ e avaliado com o resultado obitdo pelo ninho. A mudança do ninho será realizada para a melhor exploração encontrada desde a última mudança.

Para cada mudança de ninho as formigas têm suas regiões de busca apagadas da memória. Novas regiões são geradas a partir do novo ninho.

Uma formiga procura por sua presa explorando sua região de busca. Inicialmente a formiga seleciona aleatoriamente uma de suas regiões de busca u armazenadas na memória. Uma presa é obtida pela formiga se a exploração de um local u' levar a um melhor valor de f, ou seja, se $f(u) < f(u')$. A cada sucesso obtido por uma formiga, esta armazena na memória o local u' e na próxima busca por uma presa ela retornará ao local de sucesso. Para o caso de uma formiga não obter sucesso em sua exploração, na próxima busca ela deverá buscar em outra região dentre as suas regiões armazenadas na memória. Se uma região for explorada k vezes sem obter sucesso, esta é apagada da memória da formiga e uma nova região é armazenada em seu lugar. A figura 5 demonstra o comportamento de uma formiga.

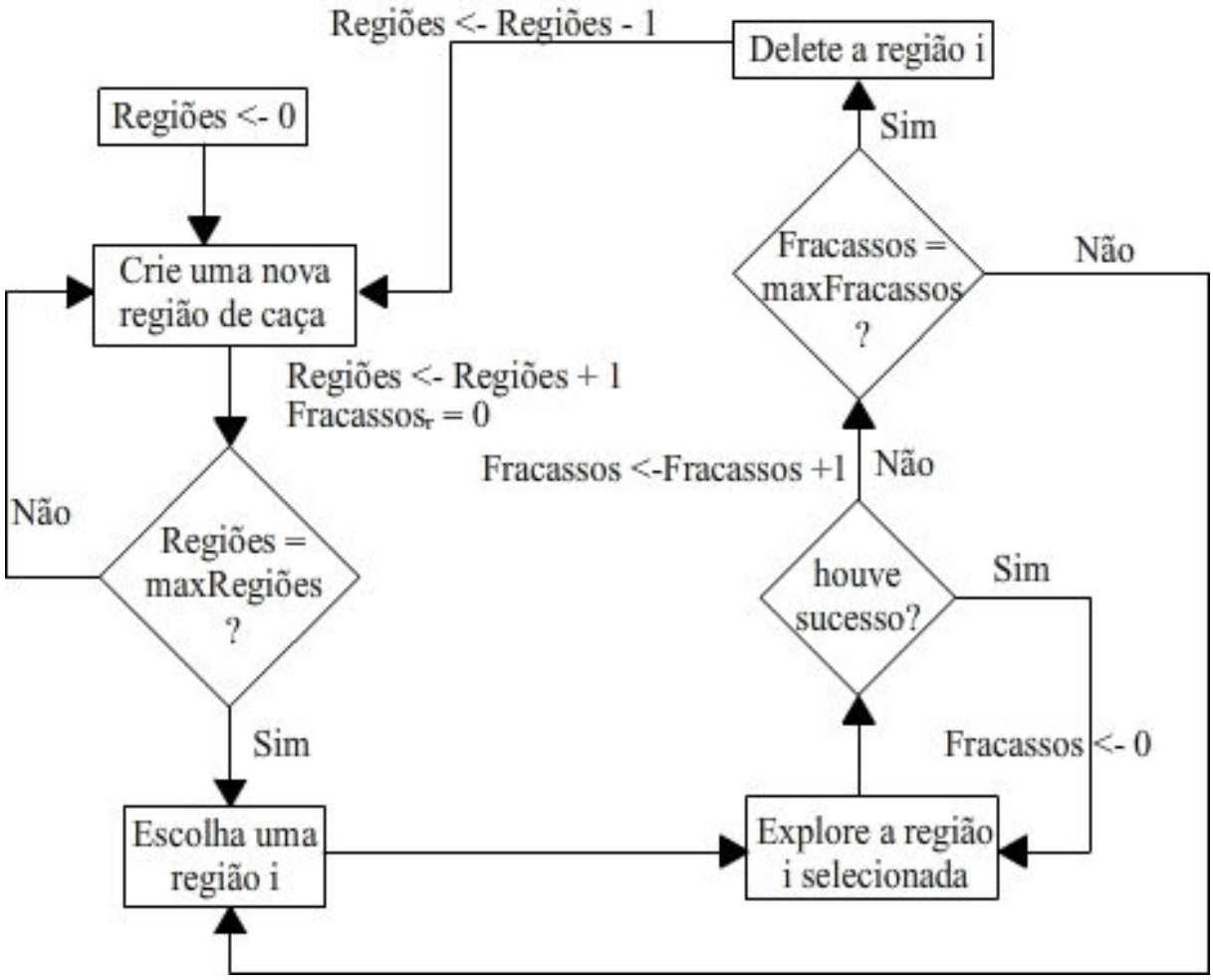

Figura 5. Esquema de comportamento de uma formiga

Uma representação em pseudocódigo do algoritmo baseado na colônia de formigas *Pachycondyla apicalis* está representado na figura 6.

1. Escolher randomicamente a localização do ninho N
2. Para cada formiga a_i, i ϵ [1 ... n]
   2.1. Se a_i tem menos que b regiões de caça na memória então crie uma nova região de caça que pertence a vizinhança de N e explore-a.
   2.2. Se não:
      2.2.1. Se a exploração anterior obteve sucesso então explore-a novamente.
      2.2.2. Se não escolha randomicamente na memória uma região para ser explorada.
3. Remova todas as regiões de busca que não obtiveram sucesso por F vezes consecutivas.
4. Se ocorreram mais de T iterações então mova o ninho e apague a memória de todas as formigas.
5. Volte ao item 2 ou pare caso o critério de parada seja satisfeito.

Figura 6. Pseudocódigo do algoritmo baseado na colônia de formigas

2.3 Sistemas Híbridos

Embora RNAs tenha se mostrado uma técnica eficiente para a solução de um grande número de problemas, é um grave erro afirmar que elas são suficientes para resolver qualquer problema de IA. RNAs apresentam vários problemas e limitações que não permitem seu uso exclusivo para a solução de uma quantidade significativa de problemas (BRAGA, CARVALHO, & LUDERMIR, 2000). Sistemas Híbridos (SHs) combinam diferentes técnicas com uma rede neural, para formar um sistema heterogêneo.

O objetivo principal no desenvolvimento de SHs é a solução de problemas de forma mais robusta e eficaz. Pois uma técnica pode conter limitações e deficiências que diminuam a performance da rede, ou até mesmo, não obtenha resultado satisfatório.

É importante ressaltar, porém, que a combinação de RNAs com outras técnicas não leva necessariamente a uma melhora do desempenho do sistema como um todo (BRAGA, CARVALHO, & LUDERMIR, 2000).

Capítulo 3

Metodologia

Este capítulo demonstra todo o processo de desenvolvimento da aplicação proposta para resolver o problema que este projeto expôs. O sistema foi implementado em *ActionScript 3.0*, no ambiente de desenvolvimento *FlashDevelop*.

A princípio, na Seção 3.1, são descritas as bases utilizadas para a obtenção dos resultados deste projeto. Na seção seguinte, a Seção 3.2, é relatado todo o processo de pré-processamento decorrido sobre os dados. Na Seção 3.3 é relatada a aplicação da Rede Neural Hibrida proposta por este projeto na classificação de dados e na previsão de dados. Na última seção desse capítulo são mostradas as técnicas utilizadas para melhorar o desempenho da rede neural construída.

3.1 Bases de Dados

A ferramenta desenvolvida neste trabalho tem como finalidade fazer tanto classificação como previsão. Para executar a classificação foi utilizada uma base de dados bastante trabalhada no meio acadêmico que é a base de dados Iris. Esta base de dados foi criada por *R. A. Fisher* e pertence ao *UCI Machine Learning Repository* desde 1988. Já para previsão foi utilizada a base de dados de vazão da usina hidroelétrica de Curuá-Una, localizada no rio Curuá-Una, no município de Santarém do estado do Pará.

A Iris é um gênero de plantas com flor, que ostentam flores de cores muito vivas. Esta base contém 3 classificações do gênero Iris e cada uma delas com 50 instâncias. Seus atributos retratam as características das flores, que são:

- Comprimento da sépala;
- Largura de sépala;
- Comprimento da pétala;
- Largura da pétala;
- Classe (Setosa, Versicolour e Virginica)

Com exceção do atributo classe, que é um atributo nominal, todos os outros são valores reais mensurados em centímetros. A Tabela 1 mostra a disposição dos dados para a base Iris.

Tabela 1. Amostra da base de dados das Flores Iris.

Comprimento Sépala (cm)	Largura Sépala (cm)	Comprimento Pétala (cm)	Largura Pétala (cm)	Classe
5.1	3.5	1.4	0.2	Iris-setosa
7.0	3.2	4.7	1.4	Iris-versicolor
6.3	3.3	6.0	2.5	Iris-virginica

A base de dados de Curuá-Una está disponibilizada no site do ONS (Operador Nacional do Sistema Elétrico) e contém as vazões médias diárias da hidroelétrica entre janeiro de 1978 e dezembro de 2007. Seu atributo de vazão é medido em metros cúbicos por segundo (m^3/s). A Tabela 2 mostra a disposição dos dados para a usina Curuá-Una.

Tabela 2. Amostra da base de dados da Usina Curuá-Una.

Data	Vazão (m^3/s)
01/01/1978	180
23/03/1987	283
14/11/1994	80
30/05/2002	282
11/09/2006	166

Porém, para que as bases descritas acima sejam utilizadas neste projeto, foi necessário que as mesmas passassem por um pré-processamento, para ficarem aptas a serem utilizadas pela ferramenta desenvolvida neste estudo.

3.2 Pré-Processamento

Como as bases de dados têm diferentes objetivos, os seus pré-processamentos se deram de diferentes formas.

Para a base Iris, que é utilizada para executar a rede em módulo de classificação, inicialmente foi necessário transformar o seu último atributo em um valor numérico, já que este se encontra na base como nominal. Como o atributo classe pode assumir 3 diferentes valores (Iris-setosa, Iris-versicolor e Iris-virginica), Tabela 1, criou-se um atributo para cada tipo de classe. Com isso a base passou a ter 3 saídas ao invés de uma. Cada classe recebeu um valor específico para seus atributos. A Tabela 3 mostra como ficou a disposição dos dados após o pré-processamento da base Iris.

Tabela 3. Base Iris após pré-processamento.

Comprimento Sépala (cm)	Largura Sépala (cm)	Comprimento Pétala (cm)	Largura Pétala (cm)	Saída 1	Saída 2	Saída 3
5.1	3.5	1.4	0.2	0	0	1
7.0	3.2	4.7	1.4	0	1	0
6.3	3.3	6.0	2.5	1	0	0

Com a base da usina Curuá-Una inicialmente foi utilizado o padrão do setor elétrico, utilizado na programação de operação do ONS e desenvolvido por especialista em hidrologia, que é aplicado para a previsão semanal de vazão. A base passou a ter 14 entradas e 7 saídas (Tabela 4), onde as 14 entradas são vazões de 14 dias passados e as 7 saídas são os 7 dias da semana que serão previstos. Além da criação da base neste padrão específico, é aplicada uma

transformação logaritmica em todos os dados da base, com o intuito de reduzir a variância da seria temporal. Isso pode melhorar o ajuste para modelos de previsão tanto em modelos estáticos quanto com as redes neurais.

Tabela 4. Base Curuá-Una após pré-processamento.

Entradas					Saídas				
n-14	...	n-2	n-1	n	n+1	n+2	n+3	...	n+7
180	...	218	266	236	267	206	241	...	212
112	...	266	236	267	206	241	153	...	210
11	...	236	267	206	241	153	172	...	258

A próxima etapa contemplou ambas as bases com o mesmo processo. Foi feita a normalização dos dados, pois a escala dos atributos deve ser levadas em consideração para evitar que a medida de proximidade seja dominada por um dos atributos. A normalização foi aplicada nos dados com o intervalo [0.1, 0.9], de acordo com a Equação 3.1.

$$y = \left(\frac{0{,}8(x_i - x_{min})}{(x_{max} - x_{min})}\right) + 0{,}1 \qquad (3.1)$$

Ao final do pré-processamento das bases, estas têm suas instâncias reorganizadas de forma aleatória para evitar a ocorrência de tendências associadas à ordem de apresentação dos dados. E por fim, a base é subdivida em 3 subgrupos: Treinamento, utilizado para a fase de treinamento da rede; Validação, utilizado para a condição de parada do treinamento; e Teste, utilizado para a verificação de desempenho ao completar o treinamento.

3.3 Modelo de Formiga

Para construir o algoritmo proposto na Seção 2.2.2 foi necessário desenvolver um modelo computacional que representasse uma formiga. Como descrito na Seção

2.2.2 uma formiga tem que explorar ao máximo o espaço de busca ao qual foi designada.

No caso estudado por este trabalho uma formiga deve representar um vetor pesos que contém todos os pesos de ligações sinápticas da rede. Este vetor de pesos constitui o espaço de busca da formiga.

Para executar a exploração a formiga deve variar cada índice do seu vetor de pesos com variação máxima definida pela variável $variacaoMaxima$ que inicialmente terá seu valor definido pela metade do raio de busca do ninho. A exploração ocorre da seguinte maneira: para cada índice do vetor de pesos da formiga será gerado um valor aleatório de variação, valor este que não pode ultrapassar o valor armazenado na variável $variacaoMaxima$. Este valor de variação será multiplicado pelo índice do vetor de pesos em questão. Ao final deste processo, um novo vetor de pesos será gerado e este será usado para executar a rede.

A formiga também é encarregada de verificar o sucesso de sua exploração. A cada ciclo da rede a formiga gera um novo vetor de pesos, a partir da exploração, que será armazenado na sua memória. A formiga tem que verificar se o EMQ (Erro Médio Quadrático) calculado com o vetor de pesos atual é melhor que o EMQ calculado no ninho e pela própria formiga. A cada sucesso encontrada o vetor de pesos é armazenado como melhor resultado. Este vetor é substituído caso haja um novo sucesso.

Antes da mudança de ninho a formiga definirá como o vetor de pesos atual aquele vetor que obtiver melhor resultado.

3.4 Inserindo Colônia de Formigas na MLP

Esta seção propõe a criação de uma rede neural híbrida combinando o algoritmo descrito na Seção 2.2.2 com a rede para o cálculo do aprendizado, ou seja, a busca dos pesos ótimos. Na Figura 6 retrata a arquitetura da MLP, pode-se observar que ela não tem um valor fixo de variáveis de entrada e de saída. Inicialmente uma formiga é definida como ninho. Para esta formiga é gerado um vetor de pesos aleatoriamente, então é executada a rede para calcular o seu EMQ. Com base nesta formiga ninho são geradas outras formigas que recebem como

regiões de busca uma variação do vetor de pesos do ninho. Para cada formiga criada, será gerado um valor de variação aleatório que não ultrapassará o valor definido na variável de variação máxima do ninho ($variacaoMax$). Este valor de variação gerado multiplicará cada índice do vetor de pesos do ninho, criando, assim, um novo vetor de pesos que será definido como o vetor de pesos da formiga em questão.

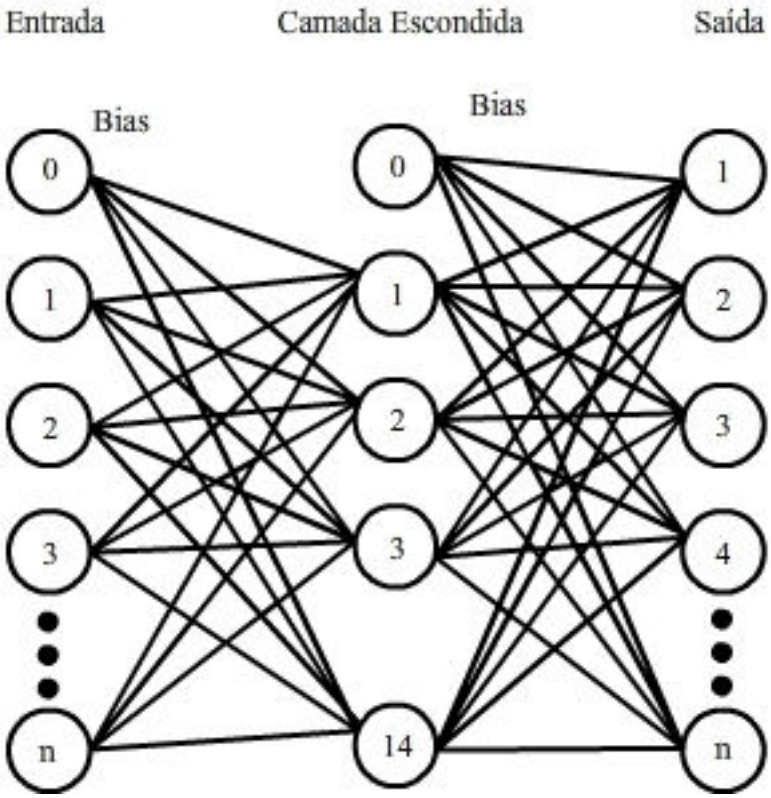

Figura 7. Arquitetura da Rede MLP utilizada

Como a rede será executada com os vetores de pesos de cada formiga, então pode-se dizer que existe uma rede para cada uma delas.

Para cada ninho estabelecido é definido um número máximo de ciclos que este deve executar, através da variável $maxCiclos$. Ao final da execução de todos os ciclos a MLP verifica qual foi a formiga de obteve melhor resultado. Em (MONMARCHÉ, VENTURINI, & SLIMANE, 2000) a critério para definir a melhor formiga depende da quantidade de sucessos que esta formiga obteve.

A MLP faz uso da validação cruzada para finalizar a execução da rede. Ao final de cada ninho a validação cruzada é executada e verifica 3 condições: Primeiro

se a rede atingiu o número máximo de mudanças de ninho; Segundo se o EMQ de validação cresce por 10 vezes consecutivamente; Por último se o EMQ de validação se manteve uniforme por 20 mudanças de ninhos consecutivas.

A mudança do ninho ocorre ao final da execução do máximo de ciclos. O novo ninho é estabelecido no novo local com o vetor de pesos da formiga que obtiver melhor desempenho. Todas as formigas têm suas regiões de busca apagadas da memória e novas regiões são geradas a partir do novo vetor de pesos do ninho.

Quando a rede atinge sua condição de término, é executado o teste de desempenho da rede. Este teste executa a rede com a formiga que obteve o melhor resultado e com os dados que foram separados para teste. O seu objetivo é verificar o desempenho da rede. Para verificar o desempenho da rede em modo de classificação é calculada a porcentagem das instâncias de dados que foram classificados corretamente, já para a rede em modo previsão é calculado EPMA (Erro Percentual Médio Absoluto). Para o cálculo do EPMA é necessário que o pré-processamento dos dados de treinamento seja revertidos, ou seja, reverter tanto a transformação logarítmica (Equação 3.2) quanto a normalização dos dados (Equação 3.3).

$$y = e^{x_i} \tag{3.2}$$

$$y = \left(\frac{(x_i - 0.1)(x_{max} - x_{min})}{0.8}\right) - x_{min} \tag{3.2}$$

O objetivo desta união de técnicas é conseguir tanto uma melhora nos resultados quanto no desempenho da MLP.

3.5 Modificações Aplicadas

Após alguns testes com a rede híbrida foi percebido que a mesma estava demorando muito para convergir. Então algumas modificações foram aplicadas tanto no modelo da formiga quanto na MLP.

Na formiga foi observado que uma única região de busca a tornaria limitada, se esta tivesse em sua memória mais de uma região de busca a sua exploração poderia ser melhor. Na criação da formiga e na mudança de ninho esta passou a

receber e armazenar em sua memória 5 regiões de busca definidas a partir da variação do ninho.

O módulo de exploração da formiga também sofreu alterações devido à nova quantidade de regiões de busca. Antes de realizar a primeira exploração, a formiga escolhe aleatoriamente uma de suas regiões e a explora. Se um sucesso for alcançado a nova exploração será feita a partir do vetor de pesos da exploração anterior. Se um fracasso for obtido a nova exploração será feita a partir da região de exploração. Se 5 fracassos consecutivos forem obtido em uma mesma região, esta região é esquecida da memória da formiga, uma nova região é inserida em seu lugar e a formiga escolhe novamente de forma aleatória qual região utilizar no seu processo de busca.

Como proposto em (VINCENTE, 2006) a redução do raio de busca nas formigas também foi implementada neste trabalho. A formiga ganhou uma nova variável ($taxaReducaoVariacao$) a qual define o valor da redução do raio de busca da formiga. essa redução ocorre da seguinte forma: sempre que a formiga conseguir um sucesso, sua próxima exploração, que ocorre a partir do vetor de sucesso, terá seu raio de busca reduzido em 10%. Esta porcentagem de redução é cumulativa, reduzindo gradativamente o raio de busca a medida que sucessos são obtidos.

Para o modelo da rede foi observado que as formigas ao receber variações do vetor de pesos do ninho poderiam causar também uma demora grande na convergência. Com o objetivo de diminuir esta lentidão definiu-se que uma porcentagem (20%) da colônia teria suas regiões de busca definidas de uma maneira diferente. Tais formigas teriam seus pesos gerados aleatoriamente entre um intervalo de [-10,10]. Com isso ficou garantido que a rede não se prenda ao espaço de busca do ninho, e se a localização do ninho estiver longe da ideal, este artifício diminuirá o tempo gasto para que o ninho se aproxime desta localização.

A avaliação da melhor formiga ao final dos ciclos de cada ninho também sofreu alterações. Com os testes foi possível observar que nem sempre a formiga que obteve mais sucessos é a formiga que obteve o menor EMQ. Como o objetivo da rede é encontrar o menor erro possível, o critério de avaliação da melhor formiga foi modificado para identificar aquela que gerou o menor erro antes da mudança do ninho.

A criação de regiões de busca para as formigas a partir do vetor de pesos do ninho também foi modificada para que a formigas ficassem mais livres dentro do espaço de busca do ninho. Ao invés de ser gerado um único valor aleatório de variação que multiplica todo o vetor de pesos do ninho para gerar uma nova região, são gerados diferentes valores aleatórios para cada índice do vetor de pesos.

Capítulo 4

Testes e Resultados

Este capítulo apresenta os resultados obtidos a partir da aplicação da metodologia detalhada no capítulo anterior. A configuração dos atributos para a simulação foram definidos no decorrer dos testes, a partir dos resultados obtidos.

A análise dos resultados para a rede em módulo de classificação foi feita através da porcentagem de acerto da classificação. Ao rodar a rede com a melhor formiga e com a base de teste é verificada a saída calculada e a saída desejada. Com o pré-processamento da base, todas as saídas da mesma foram modificadas para que seu valor fosse binário. Isto possibilitou a utilização da técnica de verificação de desempenho chamada “O Maior ganha”. Nesta técnica a saída que obtiver o maior valor será considerada como 1 e as outras como 0.

Para a previsão a análise foi baseada no EPMA, representado pela Equação 4.1, onde N_{out} é o número de neurônios na camada de saída, N é o total de instâncias da base de teste, d_{ij} é o valor desejado da previsão e c_{ij} é o valor calculado.

$$EPMA = \frac{1}{N.N_{out}} \sum_{i=1}^{N} \sum_{j=1}^{N_{out}} \left| \frac{d_{ij} - c_{ij}}{d_{ij}} \right| \quad (3.2)$$

A Seção 4.1 exibe os resultados obtidos a rede neural híbrida para classificação com a base Iris. Em seguida, a Seção 4.2 mostra os resultados para a previsão da vazão da usina Curuá-Una.

4.1 Resultados do SH para classificação

No decorrer dos testes os parâmetros que tiveram seus valores modificados foram:

- Quantidade de formigas
- Quantidade de mudanças de ninho
- Quantidade de ciclos por ninho

- Porcentagem de variação do ninho
- Porcentagem de variação na formiga
- Porcentagem de formigas livres

Foram utilizadas várias configurações na tentativa de obter melhores resultados. A figura 7 mostra a tela da aplicação desenvolvida com os parâmetros de uma simulação.

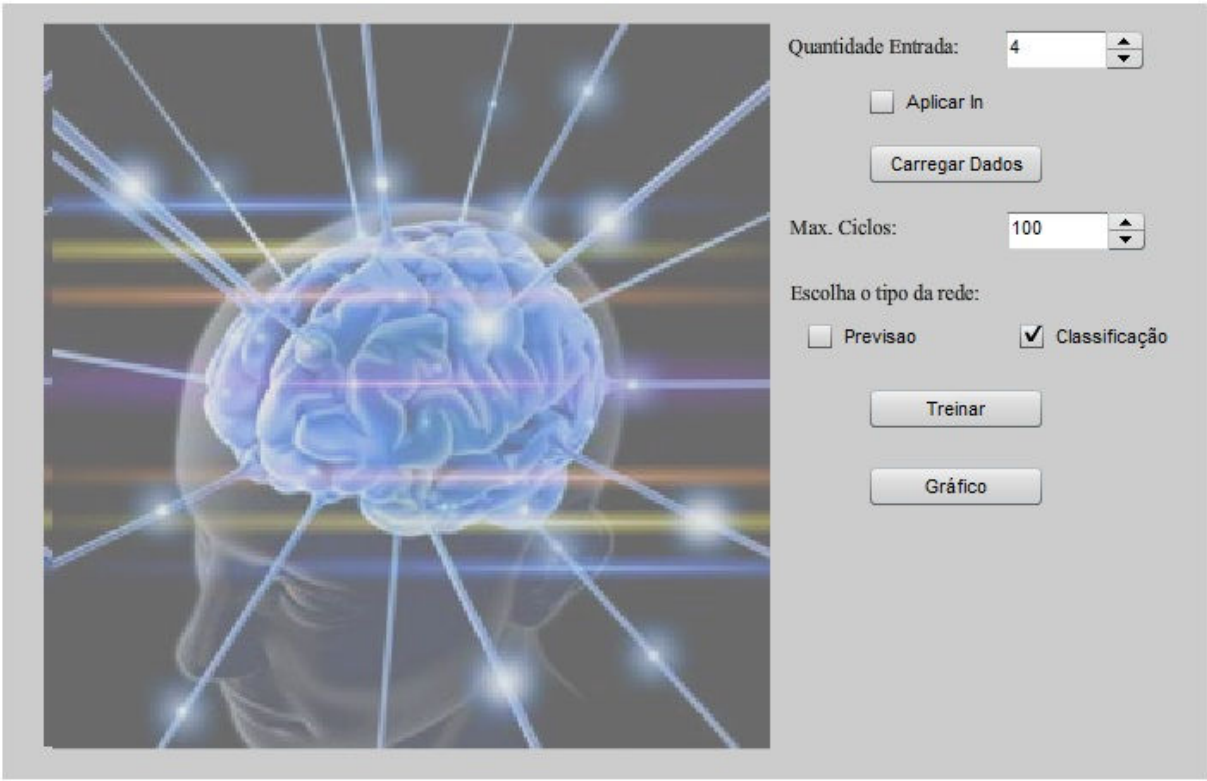

Figura 8. Tela do Aplicativo desenvolvido.

A configuração que obteve melhor resultado para a base Iris pode ser observada na tabela 5. Esta configuração obteve um percentual de acerto de 89%. A figura 8 mostra a tela da aplicação com o gráfico de validação cruzada para o melhor caso obtido. O eixo y do gráfico representa o valor do EMQ e o eixo x representa a quantidade de ninhos. Embora o percentual de classificação correta tenha atingido um valor elevado (89%), o MLP utilizando o algoritmo de aprendizagem baseado em colônia de formigas não conseguiu resultados melhores que a MLP tradicional, que utiliza o *Backpropagation* como algoritmo de aprendizagem. Em testes realizados com este MLP tradicional e a base de dados da Iris, o percentual de classificação correta atingiu 95%.

Tabela 5. Valores dos parâmetros que obtiveram o melhor resultado de classificação.

Quantidade de Formigas	5
Quantidade de ninhos	10
Quantidade de ciclos/ninho	60
Variação do ninho (%)	40
Variação da formiga (%)	15
Formigas livres (%)	20

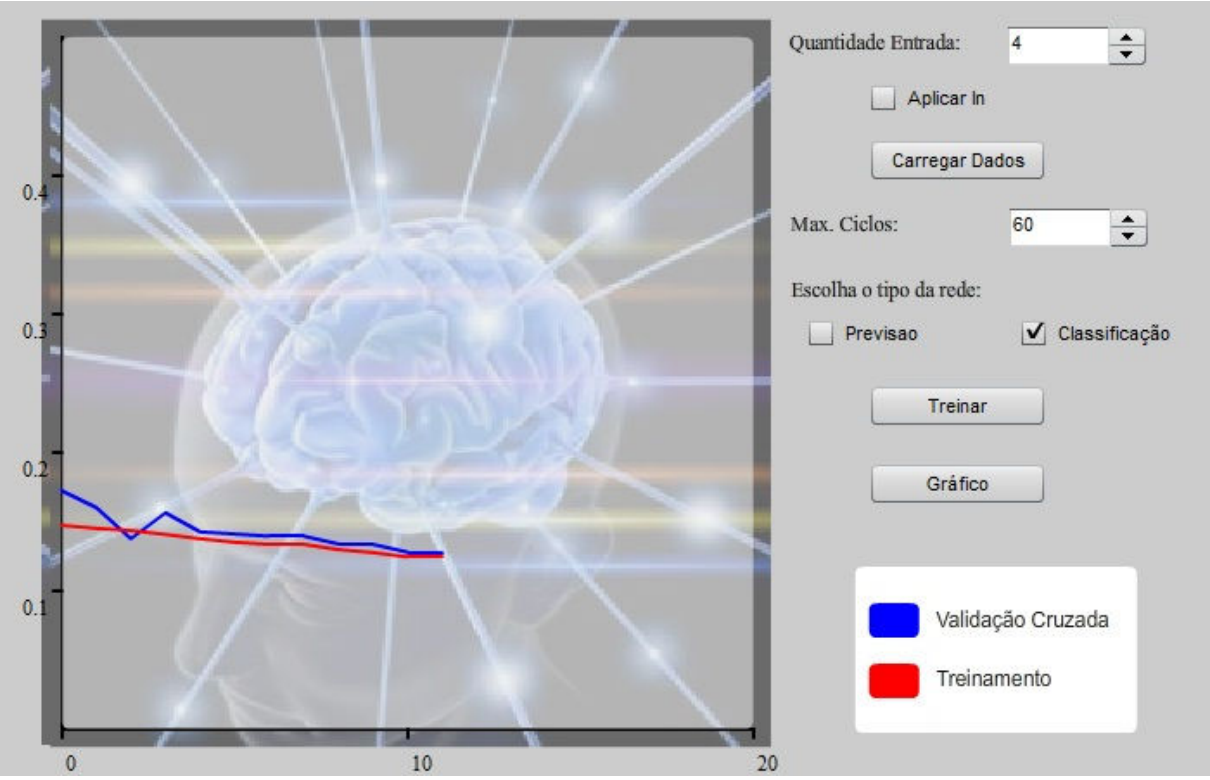

Figura 9. Gráfico de validação cruzada para o melhor caso para classificação.

4.2 Resultados do SH para previsão

Os parâmetros modificados para previsão foram os mesmos para classificação.

Como na classificação, foram utilizadas várias configurações para tentar obter o menor EPMA. As configurações para o melhor caso de previsão para a base de Curuá-Una pode ser observado na tabela 6. A configuração apresentada obteve um EPMA de 39%. A figura 9 mostra a tela da aplicação com o gráfico de validação cruzada para o melhor caso. O eixo y do gráfico representa o valor do EMQ e o eixo x representa a quantidade de ninhos.

Tabela 6. Valores dos parâmetros que obtiveram o melhor resultado de previsão.

Quantidade de Formigas	5
Quantidade de ninhos	10
Quantidade de ciclos/ninho	50
Variação do ninho (%)	40
Variação da formiga (%)	15
Formigas livres (%)	20

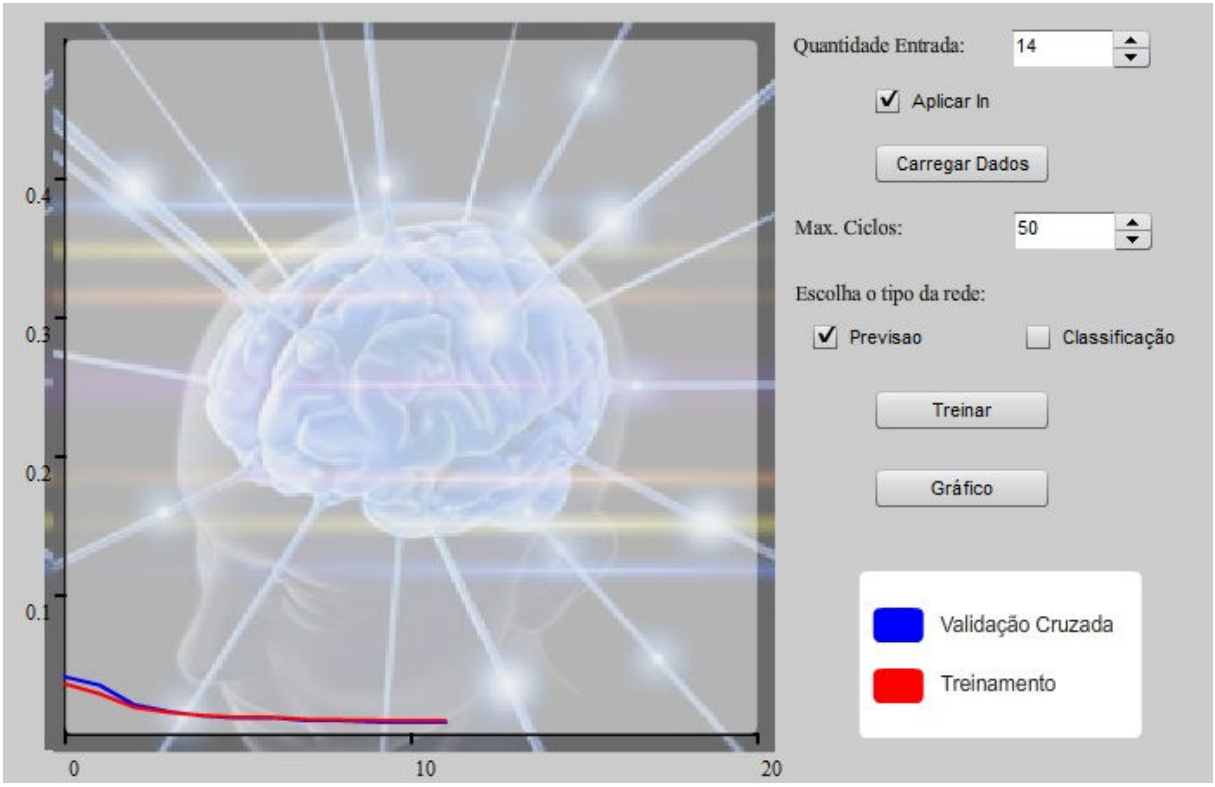

Figura 10. Gráfico de validação cruzada para o melhor caso de previsão.

Capítulo 5
Conclusões e Trabalhos Futuros

5.1 Conclusões

Esta monografia teve como principal objetivo melhorar o processo de treinamento de Redes MLPs utilizadas tanto para previsão quanto para classificação. Foi mostrado que o algoritmo de treinamento comumente utilizado, algoritmo *Backpropagation*, tem seus pontos fracos, e que o desenvolvimento de SHs visam a otimização da uma RNA.

Para tanto, foi implementado um sistema híbrido que utilizou a técnica da Rede Neural Artificial e a técnica do algoritmo de aprendizagem baseado em colônia de formigas. A avaliação da metodologia foi realizada tanto para classificação, utilizando-se a base de dados das espécies Iris, quanto para previsão, utilizando-se a base de dados da usina Curuá-Una para previsão de vazão.

Como os resultados não foram satisfatórios, algumas modificações foram implantadas no algoritmo visando melhorar o seu desempenho. Com as modificações foi melhorado o sistema de caça das formigas, deixando-as com várias regiões de busca ao invés de uma única região. No algoritmo também foi inserido um sistema de redução gradual no raio de busca da formiga, para a cada sucesso seu raio diminuir e chegar mais próximo do vetor de pesos ideal. Foram também criadas formigas livres. Estas não estavam vinculadas ao vetor de pesos do ninho, fazendo uma busca com um alcance maior que as outras. E por fim, foi modificado o sistema de avaliação da melhor formiga, para que a escolha da melhor formiga ocorresse a partir do melhor EMQ.

Após todas as mudanças implementadas, os resultados tiveram uma melhora considerável, porém ao levar em consideração resultados obtidos nas mesmas bases utilizando uma MLP com o algoritmo *Backpropagation,* não houve sucesso. Por outro lado, os resultados do algoritmo de aprendizagem proposto neste trabalho mostraram potencial. Umas das análises realizadas através dos gráficos de

validação cruzada mostra que, em geral, o algoritmo tem tendência para uma convergência prematura, o que será motivo de estudo futuro.

Uma grande dificuldade encontrada no decorrer do projeto foi o tempo de simulação, principalmente para previsão, já que a base dados continha cerca de 10 mil instâncias. O tempo da simulação dependia dos parâmetros de quantidade de ciclos e quantidade de mudanças de ninho. Alguns testes chegaram a demorar cerca de 10 horas.

Outra dificuldade enfrentada foi um problema observado na linguagem em operações com dados do tipo ponto flutuante. A linguagem utilizada, ActionScript 3.0, teve problemas em arredondar valores reais no decorrer dos cálculos. Resultados de cálculos que deveriam ser números inteiros tiveram seus valores alterados nas casas decimais.

5.2 Trabalhos Futuros

Como trabalhos futuros, tanto uma migração da linguagem quanto uma série de estudos podem ser realizados com o intuito minimizar os problemas encontrados com a linguagem ActionScript, reduzir o EPMA e aumentar o percentual de classificação correta, tais como:

- Migração de linguagem do projeto para uma linguagem mais robusta como C++ ou Java;
- Estudo de sensibilidade com relação à quantidade de ninhos e a quantidade de formigas em função da convergência prematura da rede;
- Implementar um mecanismo para definir a quantidade de camadas escondidas e neurônios de forma autmomática;
- Estudo para buscar pela configuração ideal do sistema, fazendo testes que comprovem que determinada configuração resolva a maioria dos casos e transformá-lo em uma ferramenta para usuário final que necessite do mínimo de mudanças em seus parâmetros.

Bibliografia

BRAGA, A., CARVALHO, A., & LUDERMIR, T. **Redes Neurais Artificiais: Teoria e Aplicações**. Rio de Janeiro: LTC, 2000. 262 p.

CARVALHO, M. **Uma Análise de Otimização de Redes Neurais MLP por Enxame de Partículas**. 2007. 66 f. Dissertação de Mestrado do Curso de Ciências da Computação, Universidade Federal de Pernambuco, Recife.

MONMARCHÉ, N., VENTURINI, G., & SLIMANE, M. **On how Pachycondyla apicalis ants suggest a new search algorithm**. In: Future Generation Computer Systems, 16., 2000, Tours, France. p. 937-946.

PHAM, D. T., KOÇ, E., GHANBARZADEH, A., & OTRI, S. **Optimisation of the Weights of Multi-Layered Perceptrons Using Bees Algorithm**. In: Proceedings Of 5th International Symposium On Intelligent Manufactiring Systems, 2006, Sakarya University. p. 38-46.

SAITO JUNIOR, P., NASCIMENTO JUNIOR, C., & YONEYAMA, T. **Treinamento de Redes Neurais Artificiais Utilizando Time Assíncrono**. In: IV Congresso Brasileiro De Redes Neurais, 4.,1999, São José dos Campos. p. 078-083.

VALENÇA, M. **Fundamentos das Redes Neurais**. 2 ed. Olinda: Livro Rápido,2009. 384 p.

VINCENTE, A. **O processo de Otimização Ant Systems com redução no raio de busca**. In: TEMA, 7, 2006, Cascavel. p. 159-168.