Revista Brasileira de Informática na Educação, Volume 19, Número 2, 2011

NeuroFURG: uma ferramenta de apoio ao ensino de Redes Neurais Artificiais

Carlos Alberto Barros Cruz Westhead Madsen
FURG / PPGMC - Av. Itália km 8 s/nº - Bairro Carreiros - Rio Grande - 96201-900 - Rio Grande do Sul - RS - Brasil
carlos.madsen@furg.br

Diana Francisca Adamatti
FURG / PPGMC / C3 - Av. Itália km 8 s/nº - Bairro Carreiros - Rio Grande - 96201-900 - Rio Grande do Sul - RS - Brasil
dianaadamatti@furg.br

Resumo *O presente artigo apresenta o ambiente computacional NeuroFURG, uma ferramenta voltada para a formação de estudantes de computação, tanto de graduação como de pós-graduação, na área da Inteligência Artificial conhecida como Redes Neurais Artificiais (RNAs), focando os modelos de neurônios Perceptron e Adaline. Além de permitir a construção e simulação deste dois tipos de neurônio para diferentes casos, o sistema ainda possibilita a geração de código fonte nas linguagens C, Java e PHP, permitindo que o estudante analise o código fonte e expanda a solução para um maior número de entradas ou redes com várias camadas.*

Palavras-Chave: *Inteligência Artificial, Redes Neurais Artificiais, Perceptron, Adaline, NeuroFURG, software educacional.*

Abstract *The present paper presents the computational environment NeuroFURG, a tool developed to help computer science students in their education, both in terms of graduate as well as post graduate courses, in the field of Artificial Intelligence known as Artificial Neural Networks (ANNs), focusing on the neuron models Perceptron and Adaline. Besides enabling the construction and simulation of these two types of neurons for different cases, the system still enables the conception of a source code in the C, JAVA and PHP computer languages, making it possible for the student to analyze the source code and expand the solution for a higher number of inputs or networks with several layers.*

Keywords: *Artificial Intelligence, Artificial Neural Networks, Perceptron, Adaline, NeuroFURG, Educational Software*

Recebido: 29 de Novembro de 2010 / Aceito: 29 de Janeiro de 2011 / Publicado: 24 de Agosto de 2011
DOI: 10.5753/RBIE.2011.19.02.14

1 Introdução

A Inteligência Artificial (IA) é uma ciência relativamente recente, advinda principalmente ciência da computação, tendo seu termo cunhado em 1956. Contudo, orbita pelas mais distintas áreas do conhecimento humano como filosofia, matemática, economia, neurociência, psicologia, engenharia, cibernética, lingüística, dentre outras [3].

Devido a esta pluralidade, uma definição única de IA se torna muito difícil, haja visto que este tema é abordado por diferente enfoques. No entanto, pode-se destacar duas linhas mestre, a baseada no raciocínio que tem a seguinte definição: "IA é o estudo das faculdades mentais através dos modelos computacionais." [1]; e a focada no comportamento inteligente onde: "A IA é parte da ciência da computação que compreende o projeto de sistemas computacionais que exibam características associadas, quando presentes no comportamento humano, à inteligência." [1].

A partir das definições acima apresentadas emerge o objetivo desta ciência, que vai além de se compreender os princípios que tornam a inteligência possível, mas também de ser capaz de construir entidades inteligentes [1,3]. Tendo em vista que estas entidades necessitam de um substrato físico para se materializarem, atualmente quem se vale deste papel são os computadores digitais.

Para alcançar o objetivo proposto, diversas pesquisar foram realizadas ao longo dos anos, sendo que as mesmas podem ser divididas em duas grandes famílias, a simbólica e a conexionista. A simbólica é muito próxima a tradição lógica, tendo como seus principais defensores McCarthy e Newell [1], onde o aprendizado é construído a partir de representações simbólicas de um grande número de fatos especializados sobre um domínio restrito. Sendo normalmente implementadas através de alguma expressão lógica ou árvores de decisão ou regras ou redes semânticas [2]. Já na conexionista tem-se como objetivo a modelagem da inteligência humana através da simulação dos componentes do cérebro, isto é, seus neurônios e de suas interligações, tendo sido apresentada originalmente em 1943 pelo neuropsicólogo McCulloch e pelo lógico Pitts, que propuseram o primeiro modelo matemático para o neurônio.

Este trabalho apresenta uma ferramenta de apoio ao ensino de Redes Neurais Artificiais (RNAs), denominada neuroFURG. Desta forma, esta ferramenta é embasada na abordagem conexionista.

Ferramentas para ensino de algoritmos e técnicas computacionais são bastante usuais, visto que auxiliam no aprendizado [9,10,11,12,18]. Assim, seria plausível a existência de softwares voltados para o ensino de RNAs. No entanto, não é o que se percebe, pois praticamente não existem sistemas com esta finalidade. O mais comum é a utilização de softwares científicos de âmbito geral, como o Matlab, através do módulo Netlab [19], ou ainda extensões do mesmo [20]. Uma exceção é o sistema Neural-Lab [21], projetado com uma interface visual rica que possibilita a configuração, treinamento e validação de uma RNA.

Este artigo está dividido em 5 seções. Na seção 2 são apresentados os fundamentos das redes neurais artificiais, principalmente relacionado a dois modelos, o Perceptron e o Adalaine. Na seção 3 é apresenta a ferramenta neuroFURG e suas funcionalidades. Testes realizados com a ferramenta, bem como uma análise da mesma, estão na seção 4. Finalizado, a seção 5 apresenta a conclusão e trabalhos futuros.

2 Redes Neurais Artificiais (RNAs)

As Redes Neurais Artificiais (RNAs) são a principal técnica do conexionismo, linha da IA que estuda a possibilidade de simulação de comportamento inteligente através de modelos baseados no funcionamento de cérebro humano. Dentre as principais características que distinguem este modelo do simbólico podemos destacar os seguintes [2]:

- Capacidade de aprender através de exemplos e por sua vez generalizar de forma a reconhecer instâncias similares das quais nunca havia sido apresentada;
- Bom desempenho em tarefas mal definidas, onde falta conhecimento explícito de como se encontrar uma solução;
- Tolerância a ruído, ou seja, o desempenho de uma RNA não entra em colapso na presença de informações falsas ou ausentes;
- Possibilidade de simulação de raciocínio impreciso através da associação de lógica nebulosa.

As RNAs, por sua vez, são modelos matemáticos que buscam se assemelhar as estruturas neurais biológicas, tendo a capacidade computacional adquirida por meio de aprendizado e generalização. No qual este processo de aprendizado se dá de forma iterativa onde RNA deve melhorar seu desempenho gradativamente a medida que interage com o meio externo [2].

Esta técnica de IA é aplicada principalmente me problemas de aproximação, predição, classificação, catego-

rização e otimização,como reconhecimento de caracteres, reconhecimento de voz, predição de séries temporais, modelagem de processos, visão computacional, processamento de sinais e etc [3].

2.1 Neurônio Artificial

Os modelos matemáticos de neurônios artificiais são baseados em uma simplificação do neurônio biológico. Este é formado por um corpo celular que contém o núcleo da célula, um conjunto de dendritos, por onde os estímulos são recebidos e um axônio por onde os impulsos elétricos deste neurônios são propagados para o resto da rede, como apresenta a Figura 1.

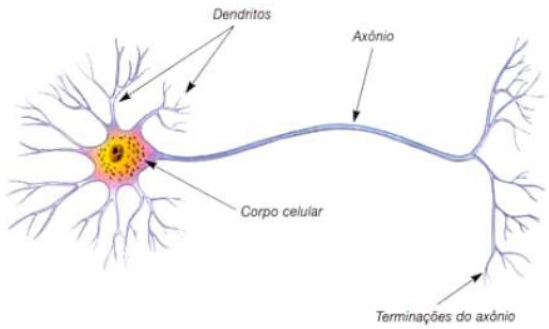

Figura 1: Simplificação do neurônio biológico [13].

A propagação do impulso elétrico através de um dendrite ou de um axônio se dá por meio da alteração da concentração de íons de k^- e Na^+. As interligações entre os neurônios ocorrem através das sinapses, pontos de contatos entre os axônios e os dendritos de dois neurônio distintos as quais convertem os impulsos elétricos e reações químicas utilizando substâncias químicas conhecidas como neurotransmissores [1,3].

2.2 Modelo de Neurônio de McCulloch e Pitts (modelo MCP)

No modelo proposto por McCulloch e Pitts o processo de informação de uma RNA é feito por meio de estruturas neurais artificiais nas quais o armazenamento e o processamento da informação são realizados de maneira paralela e distribuída por elementos processadores relativamente simples. Cada elemento processador corresponde a um neurônio artificial [2].

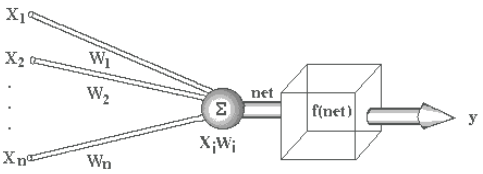

Figura 2: Neurônio artificial no modelo MCP [8].

Conforme pode-se ver pela Figura 2, cada neurônio artificial é composto por um vetor de entradas X = [x1, x2, … xn] que correspondem aos dendritos do modelo biológico, pelos quais os estímulos são captados. Ainda pode-se observar que associado a cada entrada temos um peso *wi,* o qual define o grau de conexão entre os dois neurônios, tendo um valor positivo ele, age como um excitador; caso contrário ele inibe a conexão e em última instância; caso seja zero, não existe a conexão. O núcleo da célula é implementado por uma soma ponderada das entradas pelos pesos, conhecida como net ou saída linear [1], conforme a equação 1:

$$net = \sum_{i=1}^{n} xi.wi \qquad \textbf{(1)}$$

Finalmente, após se ter obtido o net, de acordo com a Figura 2, sobre este valor ainda é submetido a uma função de ativação, f(net), a qual irá determinar qual valor (y) que o axônio irá propagar. No caso do modelo MPC ainda temos a restrição de todas as entradas xi serem binárias e a função f(net) ser um degrau, onde caso o net ultrapasse um determinado limite de disparo é propagado 1 e em caso contrário 0 [1]. Apesar de não ser mais utilizado atualmente, este modelo serviu de base para os dois modelos mais difundidos: Perceptron e Adaline.

2.3 Perceptron

O Perceptron, proposto por Rosenblatt em 1958, é muito similar ao MCP, tanto na forma de obtenção do net como por possuir uma função degrau como função de ativação, como visto na Figura 3. No entanto, apresenta a possibilidade de aprendizagem por meio de treinamento.

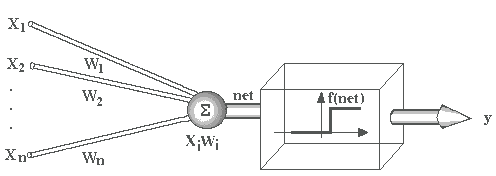

Figura 3: Modelo Perceptron [8].

Diferindo principalmente no fato de poder receber qualquer número real como entrada, em contraposição a característica binária do MPC. Tendo como principal característica ser um separador linear, ou seja, consegue distinguir entre padrões que estiverem em lados opostos de uma reta, para duas entradas, um plano, para três entradas e um hiperplano no caso de mais de três entradas [5,6].

2.4 Adaline

Proposto por Widrow em 1962, muito similar ao Perceptron, diferindo apenas no fato de não ter um função de ativação ao final, ou seja, o resultado da soma ponderada, net, é diretamente propagada para os próximo neurônios, ilustrado na Figura 4 [4].

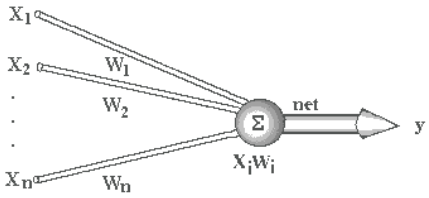

Figura 4: Modelo Adaline [8].

2.5 Aprendizado

Como foi visto anteriormente, a forma com que uma RNA aprende se dá através de treinamento, podendo este ser supervisionado ou não supervisionado.

No aprendizado não supervisionado temos um conjunto de exemplos (vetores de entrada X) que são submetidos a rede, esta por sua vez analisa cada um destes exemplos apresentados e tenta determinar se eles podem ser aglomerados de alguma maneira, formando agrupamentos ou clusters [7].

Já no aprendizado supervisionado existe a necessidade de um algoritmo de treinamento, conhecido como indutor, que tem como objetivo criar um bom classificador a partir de um conjunto de exemplos. Sendo que treinar uma RNA significa ajustar sua matriz de pesos de forma que seu resultado (vetor de saída Y) dado um exemplo (vetor de entrada X) convirja para o um certo valor esperado, conhecido como target (vetor T) [1,2].

A forma de indutor mais conhecida é a regra delta que advêm da lei de Heeb (Donald Heeb 1949) que diz "A intensidade de uma ligação sináptica entre dois neurônios aumenta se ambos são excitados simultaneamente". Considerando-se o valor esperado (target) pode-se formalizar esta lei de acordo com a equação 2 [3].

$$\Delta wi = \eta\ (ti - yi)*xi \quad \textbf{(2)}$$

Onde:

i é um índice dos vetores de entrada (X), saída (Y) e target (T)

η é a taxa de aprendizagem

xi é a entrada do neurônio associada ao peso wi

ti é o valor esperado para a entrada em i

yi é o valor resultante da propagação do neurônio dado xi

3 Ferramenta NeuroFURG

Conforme descrito anteriormente, o sistema NeuroFURG é uma ferramenta que tem como objetivo auxiliar no aprendizado de RNAs, mais especificamente a estrutura dos neurônios descritos nos modelos Adaline e Perceptron, além do algoritmo de aprendizagem, ou indutor, conhecido como regra delta. Para tanto este software se vale da interface ilustrada na Figura 5, na qual se podem observar vários itens destacados entre parênteses, sendo estes apresentados na sequência:

- **Item 1**: nesta área o estudante define com qual modelo de neurônio pretende trabalhar, se o Perceptron ou Adaline;
- **Item 2**: neste momento deve-se definir os pesos iniciais w1 e w2 relacionados, respectivamente as entradas x1 e x2 do neurônio;
- **Item 3**: caso o estudante tenha dificuldade em fazer o neurônio reconhecer os padrões ele pode se valer do *bias* ou polarizador. Este consiste em uma entrada x0 sempre em 1 e um peso w0 definido por ele, que não é modificado ao longo do treinamento. Assim o *net* descrito na equação 1 fica de acordo com a equação 3:

$$net = \sum_{i=1}^{n} xi.wi + x0.w0 \quad \textbf{(3)}$$

- **Item 4**: aqui são registrados todos os casos que serão utilizados durante o treinamento, onde para cada caso é informado as entradas x1 e x2 e seu

respectivo target, ou valor desejado, podendo ser -1 ou +1. Sendo assim, sempre serão descritos dois padrões distintos. Caso as unidades de medida das duas entradas sejam muito discrepantes, estes dados podem ser normalizados através do botão "Normalizar Casos".

Finalmente no botão "Descrições" o usuário pode documentar a que se referem as entradas. No caso da Figura 5, x1 é o comprimento do cabelo e x2 o número do sapato.

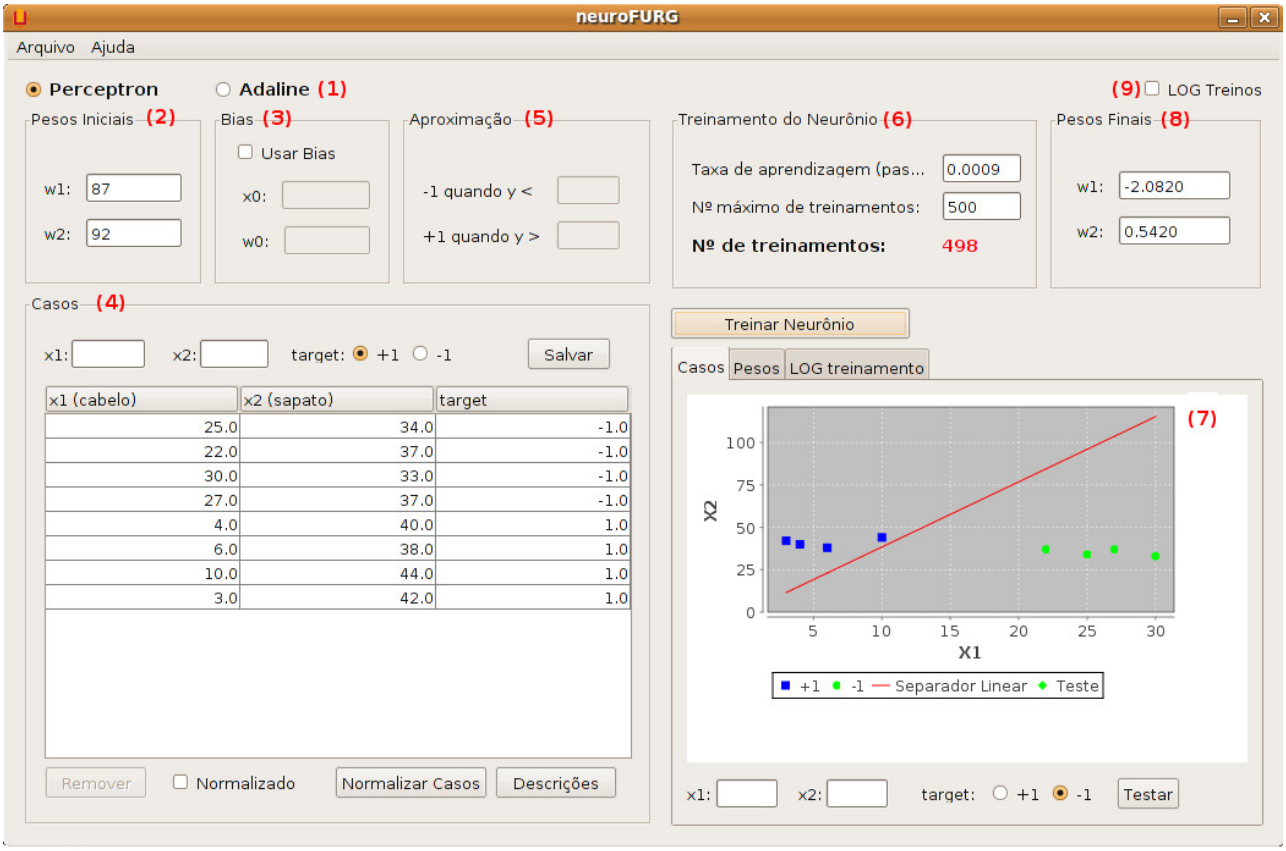

Figura 5: Ambiente NeuroFURG

- **Item 5**: esta parte do sistema fica ativa quando se seleciona o modelo de neurônio Adaline, tendo em vista que neste caso a função de ativação retorna o próprio *net*. Dificilmente a propagação da rede resultará exatamente em +1 ou -1. Sendo assim, há a necessidade de se estipular uma aproximação ou margem de erro aceitável. Por padrão, o sistema define que caso o resultado seja menor que -0,7 é considerado -1 e maior que 0,7 é considerado +1.
- **Item 6**: nesta parte da interface o estudante define qual constante de aprendizado deseja utilizar referente a equação 2. Para evitar um tempo muito grande de processamento é estipulado um critério de parada definido pelo número máximo de treinamentos que o sistema irá realizar sobre o neurônio, caso não haja convergência. E ao final, deste item temos o número de treinamentos necessário para que um dado padrão seja reconhecido com sucesso.
- **Item 7**: neste espaço cada caso do conjunto de treinamentos é plotado, sendo representado por quadrados o padrão de target +1 e com círculos o padrão com target -1. Como ilustra a Figura 6, ainda há a possibilidade de se testar o separador linear com dados que nunca foram apresentados ao

neurônio, verificando assim sua capacidade de generalização, a partir do botão "Testar". Novos exemplos de teste são plotados por losangos.

- **Item 8**: ao final de um treinamento bem sucedido, da rede os pesos finais, w1 e w2, alcançados são apresentados neste quadro.
- **Item 9**: este item permite, através do armazenamento de todos os parâmetros utilizados pelo usuário a cada treinamento, que o professor, analisando estes registros, analise o grau de capacitação alcançado pelo estudante nesta técnica de IA. Baseado nestas informações foram extraídos os resultados apresentados na seção 4.

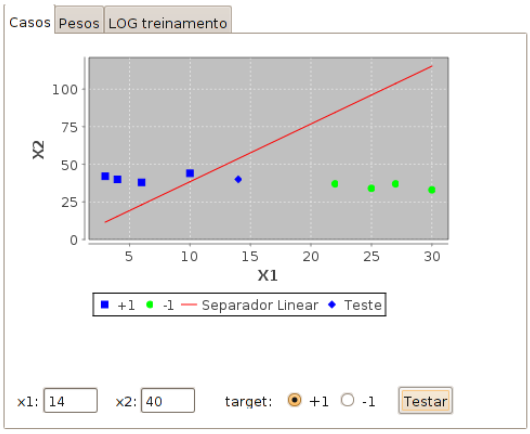

Figura 6: Testando a capacidade generalização do neurônio.

Referente ao item 7 da Figura 5, na aba "Pesos" o estudante pode verificar como foi a evolução dos pesos w1 e w2 ao longo dos diversos treinamentos realizados até houvesse a convergência do neurônio, como apresentado na Figura 7.

Ainda no item 7 da Figura 5, há a possibilidade do estudante, através da aba "LOG treinamento", analisar, para cada treinamento e caso, o algoritmo da regra delta, como ilustra a Figura 8.

Tendo em vista os itens descritos acima, destaca-se o fato de que o NeuroFURG só permite neurônios com duas entradas. O sistema foi concebido desta forma com o objetivo de que o estudante consiga visualizar o separador linear relacionado a função de ativação resultante do treinamento, visto que este pode ser plotado como uma reta em suas duas dimensões. Observando o Item 7 da Figura 5, destaca-se o separador linear representado por uma entre os clusters de quadrados e de círculos. Pode-se facilmente deduzir que a função de ativação resultante, neste caso, não distingue adequadamente os dois padrões, pois se encontra muito próxima ao cluster dos quadrados.

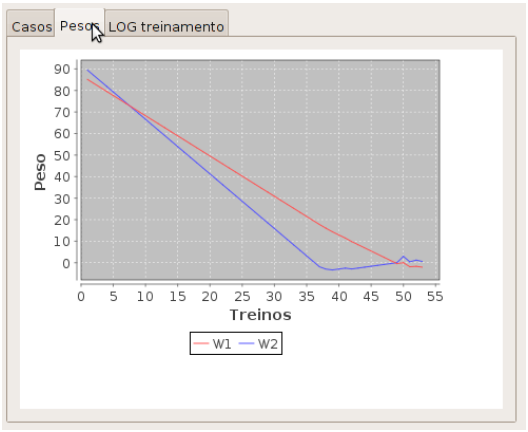

Figura 7: Evolução dos pesos ao longo dos treinamentos.

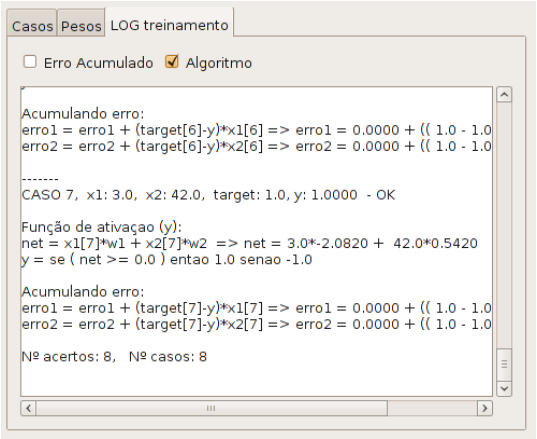

Figura 8: Passo a passo do treinamento através da regra delta.

Por fim, um diferencial do sistema NeuroFURG é a possibilidade de, uma vez o neurônio tenha reconhecido os padrões, gerar este neurônio e o algoritmo de treinamento nas linguagens C [14], JAVA[15] e PHP[16], com objetivo de que o estudante explore e modifique o código na linguagem em que estiver mais habituado. O fonte em C referente ao treinamento do neurônio ilustrado na Figura 5 é apresentado na Figura 9.

4 Testes e Resultados

Os testes do sistema foram realizados, durante a disciplina de Sistema Inteligentes do quarto ano do curso de Engenharia de Computação da Universidade Federal do

Rio Grande [17], contando com dez estudantes. Na primeira parte da aula a fundamentação teórica foi apresentada. Em um segundo momento, foi sugerido um exercício utilizando o ambiente NeuroFURG. Neste exercício foi proposto a identificação do menor número de treinamentos que seriam necessários para que um neurônio do tipo Perceptron conseguisse identificar se uma pessoa é do sexo masculino ou feminino, de acordo com o comprimento de seu cabelo (entrada x1) e o número de sapato que calça (entrada x2).

Foram apresentados os casos ilustrados no item 4 da Figura 5, onde o target -1 se refere ao sexo feminino e +1 ao masculino. Além de que foram sugeridos os pesos iniciais w1 igual a 87, w2 igual a 92 e taxa de aprendizagem sendo 0,0009 (ver item 2 e 6 da Figura 5). Dada a grande diferença de magnitude entre os pesos e a taxa de aprendizagem, inicialmente foram precisos 498 treinamentos para que fossem reconhecidos os dois padrões, número muito elevado dado a simplicidade do problema.

```c
#include <stdio.h>

#define NRCASOS 8

float w1=87;
float w2=92;

float funcao_ativacao (float x1,float x2) {
   float net =  x1*w1 + x2*w2;
   float y= -1.0;
   if (net >= 0.0) {
      y=1.0;
   }
   return (y);
}

int main () {
   int i=0;

   float passo=0.0009; //constante de aprendizado
   float x1[NRCASOS] = {25.0,22.0,30.0,27.0,4.0,6.0,10.0,3.0};
   float x2[NRCASOS] = {34.0,37.0,33.0,37.0,40.0,38.0,40.0,42.0};
   float t[NRCASOS]  = {-1.0,-1.0,-1.0,-1.0,1.0,1.0,1.0,1.0};//target

   int nr_acertos=0; //acertos por treinamento
   int nr_treinamentos=0;
   float erro1=0.0,erro2=0.0;

   while (1) {
      nr_treinamentos++;
      nr_acertos=0;
      erro1=0.0;
      erro2=0.0;

      for (i=0;i<NRCASOS;i++) { //treinamento
         float y = funcao_ativacao (x1[i],x2[i]);

         if (y==t[i]) { //comparando y com o target
            nr_acertos++;
         }

         erro1+=(t[i]-y)*x1[i];
         erro2+=(t[i]-y)*x2[i];
      }//fim for
      printf ("Treino: %d, acertos: %d\n",nr_treinamentos,nr_acertos);

      if (nr_acertos==NRCASOS) { //convergiu
         break;
      }
      else { //atualizando os pesos
         w1=w1+passo*erro1;
         w2=w2+passo*erro2;
      }
   }//fim while


   printf ("Pesos Finais: w1: %f, w2: %f, \n",w1,w2);
}
```

Figura 9 : Código em C para o treinamento do neurônio Perceptron.

Durante o exercício, os estudantes foram muito participativos, não apresentando dificuldades no entendimento da ferramenta, pois realizaram diversos testes e a cada solução proposta, discutiam sobre os resultados encontrados com os colegas. Assim, o processo de ensino aprendizagem ocorreu de forma empírica, baseada em tentativas-erro, colocando em prática os conhecimentos teóricos anteriormente apresentados.

Pode-se observar na Figura 10, que todos os estudantes conseguiram chegar a um número de treinamentos significativamente baixos, tendo em conta os parâmetros iniciais propostos.

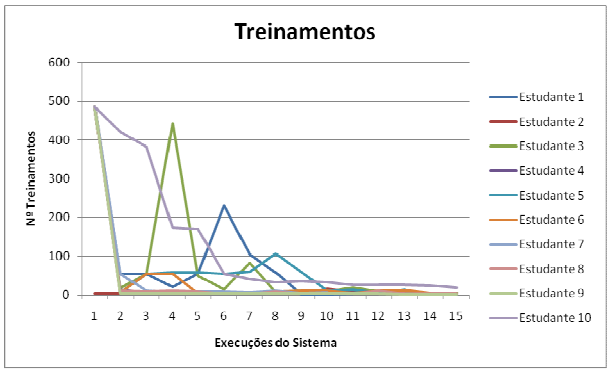

Figura 10 : Gráfico de Nº de Treinamentos X Execuções do Sistema

Todavia, ao se observar a Figura 11 e a Figura 12 que ilustram a variação dos pesos iniciais w1 e w2, respectivamente, ao longo da utilização da ferramenta e fazendo um contraponto com a Figura 13 que apresenta a variação da taxa de aprendizagem, fica claro que as estratégias para se conseguir o número de treinamentos reduzidos foram as mais variadas. Pode-se destacar os estudantes que praticamente só modificaram a taxa de aprendizagem, deixando os pesos quase que inalterados; e outro grupo que fez pequenas alterações na taxa de aprendizagem e mudanças radicais nos pesos.

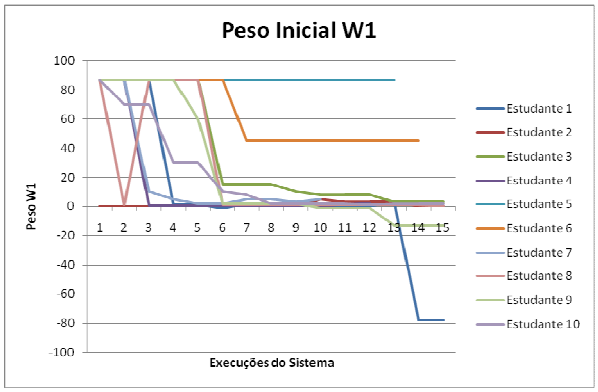

Figura 11 : Gráfico do Peso Inicial w1 X Execuções do Sistema

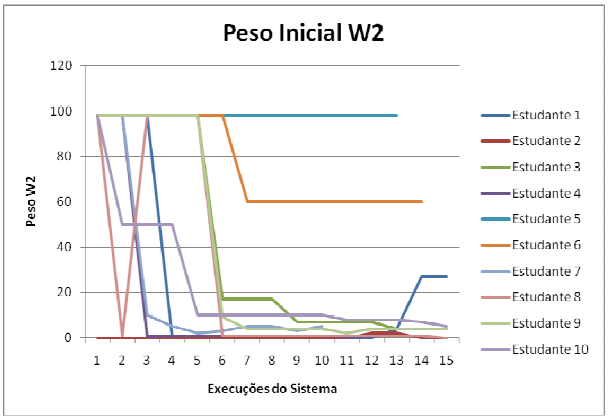

Figura 11 : Gráfico do Peso Inicial w2 X Execuções do Sistema

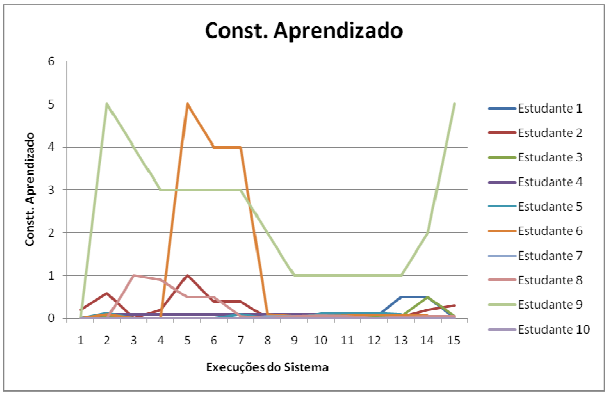

Figura 13 : Gráfico da Constante de Aprendizado X Execuções do Sistema

Para ilustrar melhor estas duas estratégias predominantes, vamos destacar a atividade do estudante 5 que optou em trabalhar somente com a taxa de aprendizagem. As Figuras 14, 15, 16 e 17 apresentam, respectivamente, o número de treinamentos, constantes de aprendizado, pesos iniciais e pesos finais nos quais o Perceptron convergiu.

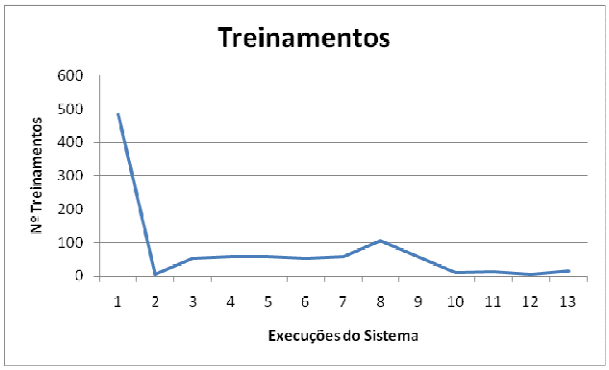

Figura 14 : Gráfico de Nº de Treinamentos X Execuções do Sistema para o Estudante 5

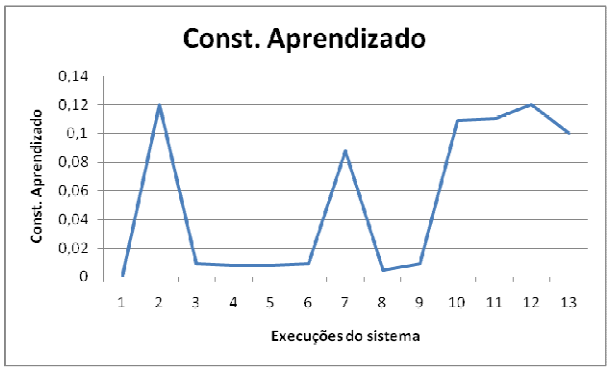

Figura 15 : Gráfico da Constante de Aprendizado X Execuções do Sistema para o Estudante 5

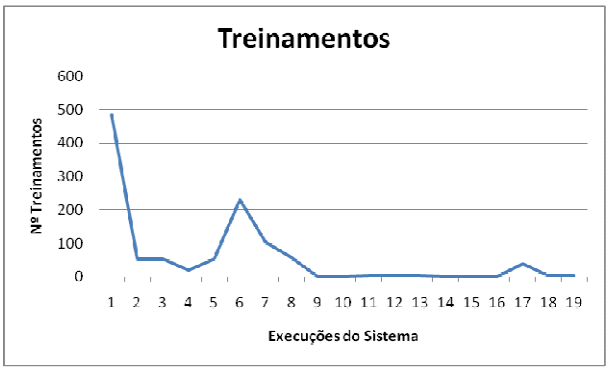

Figura 18 : Gráfico de Nº de Treinamentos X Execuções do Sistema para o Estudante 1

Figura 16 : Pesos iniciais w1 e w2 X Execuções do Sistema para o Estudante 5

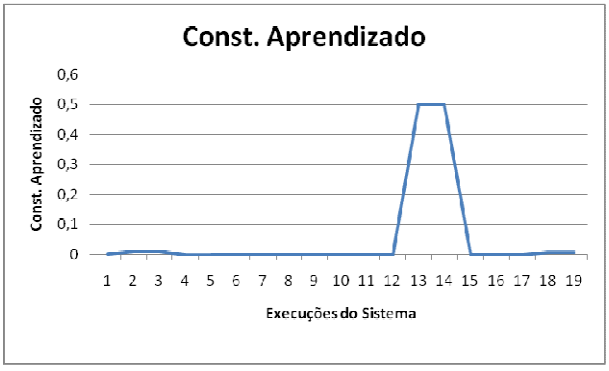

Figura 19 : Gráfico da Constante de Aprendizado X Execuções do Sistema para o Estudante 1

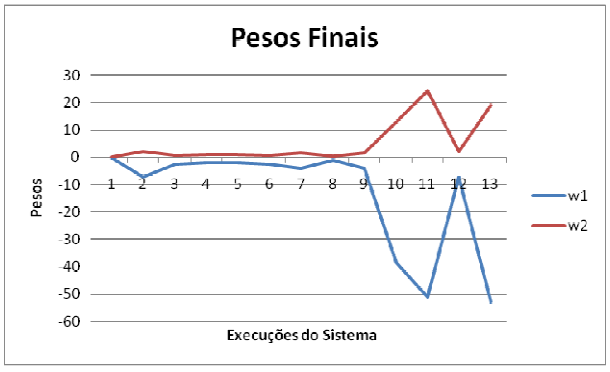

Figura 17 : Pesos finais w1 e w2 X Execuções do Sistema para o Estudante 5

Figura 20 : Pesos iniciais w1 e w2 X Execuções do Sistema para o Estudante 1

Se valendo de uma estratégia oposta, apesar de não ser tão radical a ponto de não modificar a constante de aprendizado, mas com alterações nos pesos inicias, tem-se o estudante 1, as Figuras 18, 19, 20 e 21 ilustram, respectivamente, seu número de treinamentos, constantes de aprendizado, pesos iniciais e pesos finais.

Figura 21 : Pesos finais w1 e w2 X Execuções do Sistema para o Estudante 1

Tendo todos os estudantes encontrado os pesos iniciais e constantes de aprendizado que os satisfizessem, foi proposto um segundo exercício onde eles deveriam gerar um fonte em C dos seus Perceptrons, através da ferramenta. Este código é similar ao apresentado na Figura 9 diferindo apenas nos valores iniciais das variáveis w1, w2 e passo. Eles deveriam expandir o neurônio para que aceitasse mais um entrada, a altura da pessoa, utilizando casos apresentados na Tabela 1.

Cabelo (cm)	Nº Sapato	Altura (m)	Sexo
25	34	1,55	-1
22	37	1,7	-1
30	33	1,65	-1
27	37	1,68	-1
4	40	1,75	1
6	38	1,7	1
10	44	1,9	1
3	42	1,85	1

Tabela 1: Casos para treinamento do Perceptron modificado

Ao final da aula, todos conseguiram modificar o fonte, compilar e fazer com que seus neurônios aprendessem os dois padrões a partir das três entradas. Tendo em vista que todos os estudantes tiveram competência de interpretar e modificar o código, fica evidente que os mesmos foram capacitados, no que tange o modelo de neurônio Perceptron e ao algoritmo de aprendizagem baseado na regra delta.

5 Conclusões e Trabalhos Futuros

As RNAs são técnicas de IA consolidadas, tanto na área acadêmica quanto no mercado, tendo aplicações que variam desde o reconhecimento de caracteres até a predição de cotações de ações na bolsa de valores [3]. Contudo os modelos de neurônio utilizados no sistema, Perceptron e Adaline, são caracterizados por se restringirem a problemas linearmente separáveis, quando configurados em redes de uma única camada. Para problemas mais complexos há a necessidade de redes com camadas ocultas [1].

O NeuroFURG se destaca pela facilidade com que o estudante consegue configurar, treinar, validar e gerar o código fonte do neurônio artificial. A configuração se dá pela definição dos valores dos pesos, taxa de aprendizado e casos de treinamento. Tantos treinamentos quanto forem necessários podem ser feitos simplesmente pressionando o botão "Treinar Neurônio", uma vez que todos os casos tenham sido registrados. A validação do treinamento realizado, ou seja, se o separador linear obtido com treinamento é uma boa solução ou não para o problema, resume-se a observar a linha plotada entre os dois clusters. Finalmente caso o estudante queira conhecer mais profundamente o modelo, ou se houver necessidade de expandir a solução, ele pode optar por gerar o fonte de neurônio nas linguagens C, JAVA e PHP.

Durante os testes realizados na disciplina de Sistemas Inteligentes, todos os estudantes conseguiram diminuir consideravelmente o seus tempos de treinamentos, mesmos se valendo de estratégias distintas. E também conseguiram expandir a solução apresentada de duas para três dimensões. Com isso, pode concluir que as funcionalidades implementadas no ambiente podem ser executadas de forma eficiente, auxiliando no processo inicial de aprendizagem de RNAs. Em virtude desse desempenho, houve um grande interesse por parte dos docentes ligados a IA na FURG em adotar este software em suas disciplinas. Dado as possibilidades vislumbradas pela utilização do sistema, novas necessidades vieram a tona, sugerindo a continuação do trabalho no que se refere ao desenvolvimento de um software mais avançado, que possibilite a criação de redes com camadas ocultas e que abranja outros modelos de neurônio.

A fim de avaliar mais profundamente a utilização do ambiente, como trabalhos futuros, novos testes serão realizados, também a nível de pós-graduação, com alunos do Mestrado em Modelagem Computacional da FURG, na disciplina de Elementos da IA, que recebe alunos de diversas áreas do conhecimento, como matemática, estatística, física, além de alunos de computação. Nesses testes, pretende-se avaliar a eficiência do sistema de uma forma quantitativa, onde um grupo utilizará o sistema e outro não. Desta forma, será possível comparar o nível de aprendizado entre os grupos e a eficiência do sistema.

6 Agradecimentos

Os autores agradecem a todos os estudantes que utilizaram o software em sala de aula e contribuíram com preciosas sugestões. Ao técnico de laboratório Msc. Glauco Delevedove, que prontamente deixou disponível o NeuroFURG em todos os laboratórios do Centro de Ciências Computacionais. Ao Prof. Dr. Adriano Velasque Werhli, pelo incentivo e sugestões. E finalmente a Profa. Dra. Sílvia Silva da Costa Botelho que gentilmente cedeu parte de sua aula para que o testes fossem realizados.

7. Referências

[1] G. Bittencourt. Inteligência Artificial - Ferramentas e Teorias. Ed. da UFSC. 3º ed. Florianópolis SC, 2006.

[2] S, Rezende et al. Sistemas Inteligentes - Fundamentos e Aplicações. Ed. Manole. Barueri SP. 2005.

[3] S, Russell; P, Norving. Inteligência Artificial. Ed. Campus. 2ª Ed. Rio de Janeiro RJ 2004.

[4] B. Widrow, M. Lehr. Perceptron, Adalines, and Backpropagation. In *The Handbook of Brain Therory and Neural Networks*. MIT Press. páginas 719-724, 1995.

[5] H. D. Block. The Perceptron: A Model for Brain Functioning. *Reviews of Modern Physics*. vol. 4 nr. 1, páginas 123-135, 1962.

[6] F. Rosenblat. The Perceptron: A probabilistic model for information storage and organization in the brain. *Reviews of Modern Physics*. vol. 65 nr. 6, páginas 386-408, 1958.

[7] R. Hanson, J. Stutz, P. Cheeseman. Bayesian Classification Theory. Technical Report FIA-90-12-7-01, NASA Dezembro 1990.

[8] C. Y. Tatibana, D. Y. Kaetsu. Uma Introdução as Redes Neurais. Disponível em http://www.din.uem.br/ia/neurais/. Acesso Novembro/2010.

[9] R. Santiago, R. L. S. Dazzi. Ferramenta de apoio ao ensino de algoritmos. Anais do XII Seminário de Computação (SEMINCO). Ed. da FURB. páginas 79-86, 2004.

[10] C. E. Klock, R. P. Ribas, A. I. Reis, Karma: um ambiente para aprendizado de síntese de funções Booleanas. Revista Brasileira de Informática na Educação. Vol. 18 nr. 2, páginas 33-42, 2010

[11] P. V. Vieira, A. L. Raabe, C. A. Zeferino, Bipide - Ambiente de Desenvolvimento Integrado para a Arquitetura dos Processadores BIP. Revista Brasileira de Informática na Educação. Vol. 18 nr. 1, páginas 32-43, 2010

[12] R. P. Santos, H. A. X. Costa. Um software gráfico educacional para o ensino de algoritmos em grafos, International Associaton for Development of the Information Society. páginas 358-362, 2006.

[13] L. Packer. Neurociência - Elementos de Neurofisiologia Farmacologia Psiquiatria. Disponível em: http://www.filosofiaadistancia.com.br/gravações agosto/Neurociências/Neurociência - apostila I.htm Acesso Novembro/ 2010.

[14] B. W. Kernighan. C: A Linguagem de Programação. Ed. Edisa. Porto Alegre RS, 1986.

[15] C. S. Horstmann et al. Core JAVA 2 - Fundamentos. vol 1. Ed. Makron Books. São Paulo SP, 2001.

[16] W. Hughes. PHP Guia do Desenvolvedor. Ed. Berkeley Brasil. São Paulo SP, 2001.

[17] Universidade Federal do Rio Grande. Disponível em http://www.furg.br. Acesso Novembro/ 2010.

[18] J. E. R. Queiroz, P. T. Firmino Jr, A. C. Hora, V A Porto, SImPLe: uma Ferramenta de Suporte ao Processo de Ensino-Aprendizagem de Processamento Digital de Imagens. Revista Brasileira de Informática na Educação. Vol. 15 nr. 2, páginas 1-13, 2007

[19] I. T. Nabney. NETLAB Algorithms for Pattern Recognition. Ed. Springer. London, England, 2004.

[20] E.G. Roselló, J. B. Pérez-Schofield, J. G. Dacosta, M. Péres-Cota, Neuro-Lab: A highly reusable software-based environment to teach artificial neural networks. Computer Applications in Engineering Education. Vol. 11, nr. 2, páginas 1002-1042 , 2003.

[21] R. B. Araújo, P. L. Melo, Protótipo de Software para Aprendizagem de Redes Neurais Artificiais. Anais do XXXIV COBENGE: Congresso Brasileiro de Educação em Engenharia. páginas 1435-1445 , 2006.