

**COPPE/UFRJ**

**CPE 721 - Redes Neurais Feedforward**  
**Prof.: Luiz Calôba**

PROGRAMA PARA DEMONSTRAÇÃO DO PROCESSO DE APRENDIZADO DOS NEURÔNIOS DA CAMADA INTERMEDIÁRIA DE UMA REDE NEURAL FEEDFORWARD DE 2 CAMADAS, COM UMA VARIÁVEL DE ENTRADA E UMA VARIÁVEL DE SAÍDA.

**OBJETIVO:** Será montada uma rede neural de 2 camadas, com uma variável de entrada e uma variável de saída. Após o processo de treinamento da rede através do método “error backpropagation”, deverá se verificar a contribuição de cada neurônio da camada intermediária para a composição do mapeamento entrada-saída.

**ALUNOS:** Marcos José Carvalho de Mello  
José Lorenzo Paz

Agosto / 2004

## 1. INTRODUÇÃO

“Realmente, o vislumbre de um software rodando em um computador que emule certas condições que ocorrem em um cérebro humano é algo fascinante. Talvez uma visão utópica seria a de um sistema emulando as atividades principais do cérebro. Assim, teríamos robôs que pensariam como seres humanos, executando tarefas insalubres, tornando a vida em uma sociedade tecnológica bem diferente em vários aspectos, semelhante ao que vemos na expressão da ficção científica. Talvez levem centenas, milhares ou milhões de anos para vermos coisas como estas acontecendo, porém, isto é assunto que gera muita controvérsia na comunidade científica nos dias de hoje e nos que virão” [Medeiros 2003].

É desta forma que vemos o assunto “Redes Neurais”, como algo fascinante a ser descoberto por nossas mentes inquiridoras. E, com este mesmo fascínio, aceitamos a tarefa de dar esta contribuição que, por se tratar de um trabalho de final de período, será apenas uma gota d’água neste vasto oceano.

## 2. PROPOSTA DO TRABALHO

Como proposta de desenvolvimento deste trabalho, iremos apresentar um programa que demonstre em uma rede neural do tipo feedforward de 2 camadas, com apenas uma variável de entrada e uma variável de saída, como os neurônios da camada intermediária contribuem para a representação do mapeamento entrada-saída.

## 3. ESTRUTURA DA REDE

A estrutura da rede utilizada na implementação do programa é mostrada na figura 1.

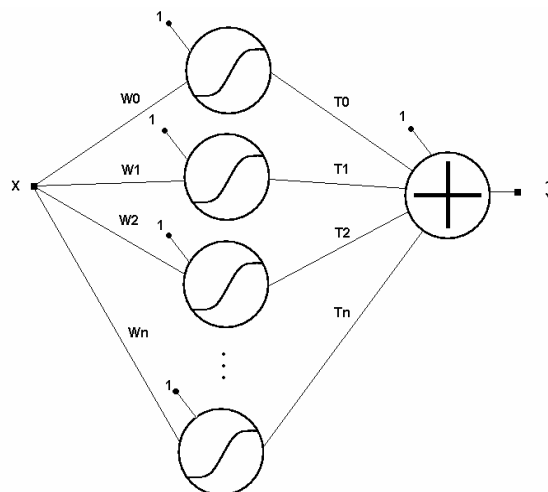


Figura 1 – Estrutura da Rede

Nesta estrutura verificamos que temos um neurônio do tipo linear na camada de saída e um número  $n$  de neurônios do tipo tangente hiperbólico na camada intermediária.

O treinamento a ser utilizado será o backpropagation por regra delta.

## 4. IMPLEMENTAÇÃO

Por sugestão do professor, o programa deveria ser apresentado na forma de um pacote “fechado”, possibilitando sua utilização sem a necessidade de instalação de programas adicionais.

Para isso foi escolhida a linguagem Delphi, que é uma implementação da linguagem Pascal com orientação a objetos oferecida pela Borland na forma RAD (Rapid Application Development).

Utilizamos a versão 7, porque desejávamos mudar dinamicamente o tamanho da matriz que armazena as informações dos neurônios da camada intermediária, conforme a decisão do

usuário. Em versões anteriores, seria necessária a utilização de ponteiros e listas ligadas, mais suscetíveis a erros de programação.

## 5. UTILIZAÇÃO DO PROGRAMA

Para rodarmos o programa, copiamos o arquivo executável do

disquete para o disco rígido, juntamente com os arquivos de treinamento. O programa foi compilado para funcionar em qualquer computador rodando o sistema operacional Windows 95 ou superior.

Ao executarmos o programa, será mostrada a tela indicada na figura 2.

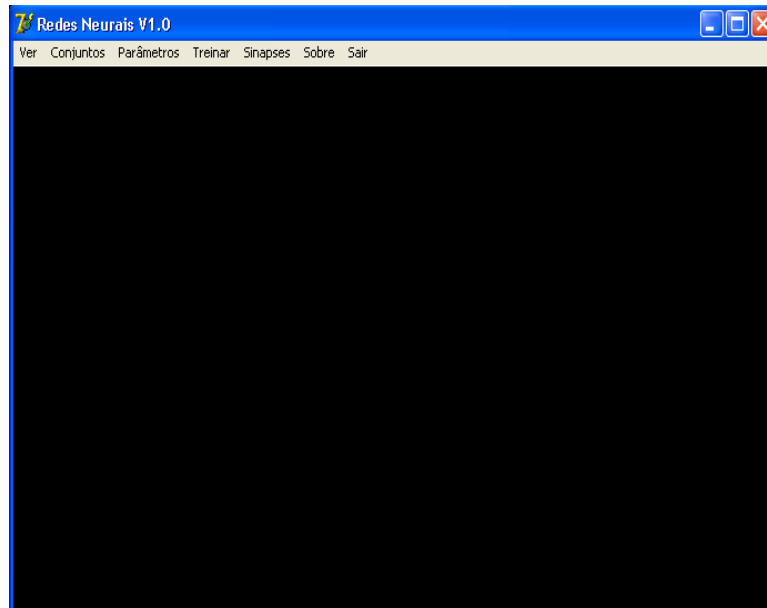


Figura 2 – Tela inicial do programa

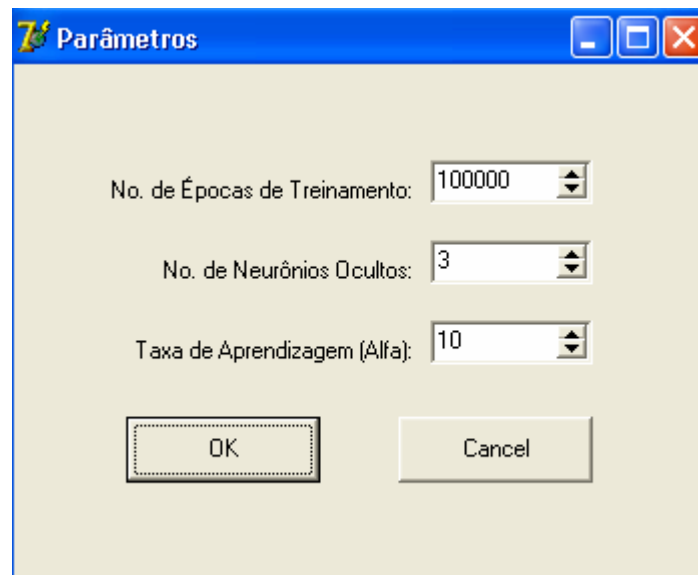


Figura 3 – Seleção de Parâmetros

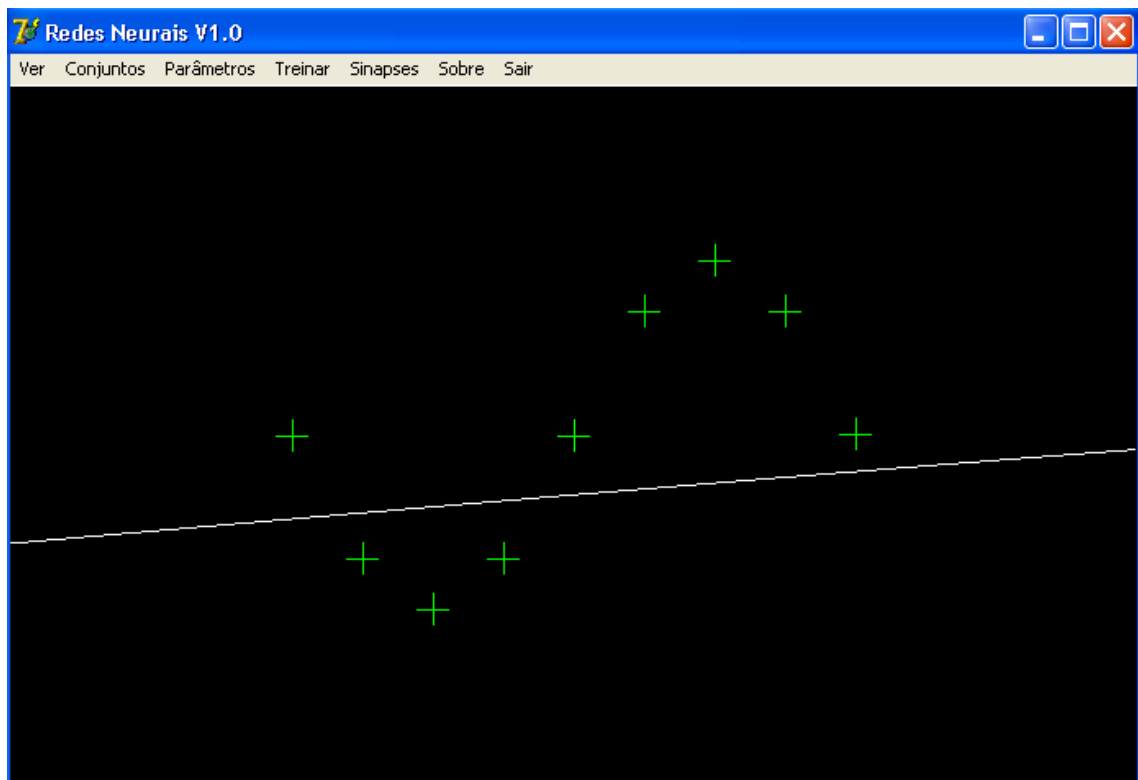
Clicando em “Parâmetros”, temos a tela mostrada na figura 3.

Nela podemos definir o número de épocas de treinamento, o número de neurônios da camada intermediária e a taxa de aprendizado. O número de épocas de treinamento representa o número de vezes que serão executados cada passo “feedforward”, “backpropagation” e correção das sinapses utilizando regra delta. A taxa de aprendizagem é o parâmetro  $2 \cdot \alpha$  (multiplicado por 100 para permitir o uso do componente SpinEdit do Delphi), utilizado para a correção das sinapses durante a “descida” do gradiente.

O próximo passo é selecionar um dos conjuntos de treinamento. Clicamos em “Conjuntos” e, na tela que se abre “Selecionar conjunto de treinamento”. Escolhemos um dos conjuntos e em seguida “OK”. A tela mostrada será como a da figura 4.

Os pontos em verde representam os pares de entrada-saída fornecidos pelo conjunto de treinamento.

Já a linha em branco é a função da saída pela entrada antes do treinamento (os valores iniciais das sinapses, incluindo os bias dos neurônios, são inicialmente sorteados para assumirem valores entre -0.5 e +0.5).



**Figura 4 – Seleção do conjunto de treinamento Sen.trn**

Neste ponto podemos “ensinar” à rede mapear a função clicando em “Treinar”. Serão feitas 100000 iterações (ou conforme o número de épocas de

treinamento definido no menu “Parâmetros”) para correções das sinapses na direção do erro mínimo. Vale ressaltar que para este programa,

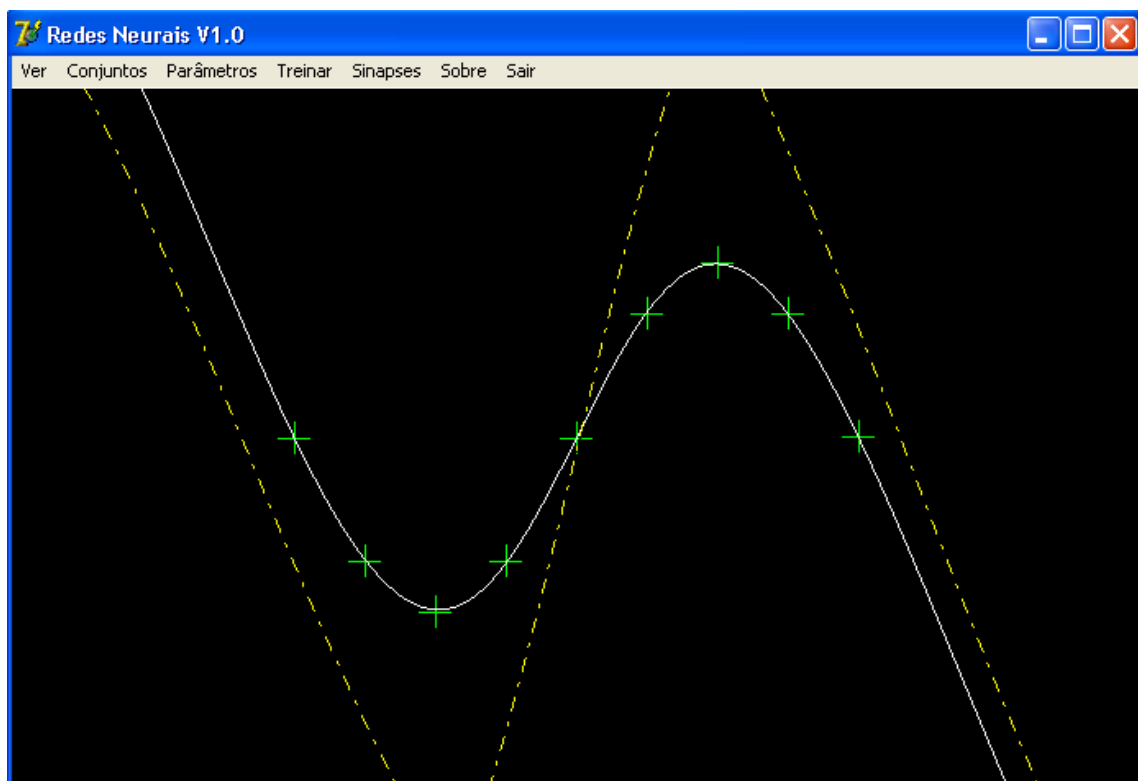
ao contrário de outras aplicações onde um conjunto de teste é fornecido, não há nenhum controle de “overtraining”, já que estamos apenas preocupados na forma como os neurônios ocultos contribuem para representar a curva de entrada-saída.

Se o “chute inicial” foi dado de tal forma a permitir que cheguemos ao mínimo global ou, pelo menos a um mínimo local aceitável, veremos a curva em branco passando bem próxima aos

pontos fornecidos pelo conjunto de treinamento.

Clicando em “Sinapses”, podemos ver os valores das sinapses que representam o mapeamento atual da rede.

Selecionando o menu “Ver” – “Neurônios Ocultos”, teremos em amarelo tracejado o mapa entre entrada-saída dado por cada um dos neurônios da camada intermediária. Para nosso exemplo, isto pode ser visto na figura 5.



**Figura 5 – Rede treinada mostrando a contribuição dos neurônios da camada intermediária em amarelo.**

## 6. SUGESTÕES DE MELHORIAS

Por se tratar de trabalho realizado com tempo reduzido, alguns itens menos importantes não foram implementados para que o projeto ficasse pronto dentro do prazo estipulado.

Tais implementações, acreditamos, contribuiriam para deixar o programa

mais “completo” do ponto de vista didático, dentro da disciplina de Redes Neurais.

Em princípio, a inclusão dos respectivos conjuntos de teste, bem como a demonstração das curvas de erro, permitiriam o controle de “overtraining” da rede.

Além disso, tornar o valor inicial máximo (em módulo) dos valores das

sinapses configurável pelo usuário, traria mais liberdade para se evitar mínimos locais (e conseqüentemente uma possível paralisia no treinamento).

Outra possível implementação seria a configuração dos valores máximos e mínimos dos pares de entrada e saída. Atualmente só devem ser utilizados conjuntos de treinamento cujos pares estejam no intervalo  $[-1, +1]$ .

Finalmente, como todo programa recém compilado, será necessário corrigir os bugs que irão surgir durante a sua utilização.

## 7. CONCLUSÃO

No apêndice A, foram selecionadas algumas situações onde fica bastante evidente a contribuição da camada oculta para o mapeamento da rede. Podemos observar, como já era esperado, que cada neurônio acaba sendo “responsável” por aproximar cada pedaço da curva por uma tangente hiperbólica cujos parâmetros são determinados pelos valores das sinapses ( $w_i$  e  $t_i$ ) e do bias (vide novamente a figura 1).

Notamos também, em alguns casos, a existência de neurônios que quase não contribuem para o funcionamento da rede, sendo considerados fortes candidatos ao “prunning” (poda). Dá para ver que as sinapses associadas a estes neurônios ( $w_i$ ,  $t_i$  e bias) têm valores quase nulos.

## 8. BIBLIOGRAFIA

Medeiros, Luciano Frontino de – *Redes Neurais em Delphi*. Visual Books, 2003.

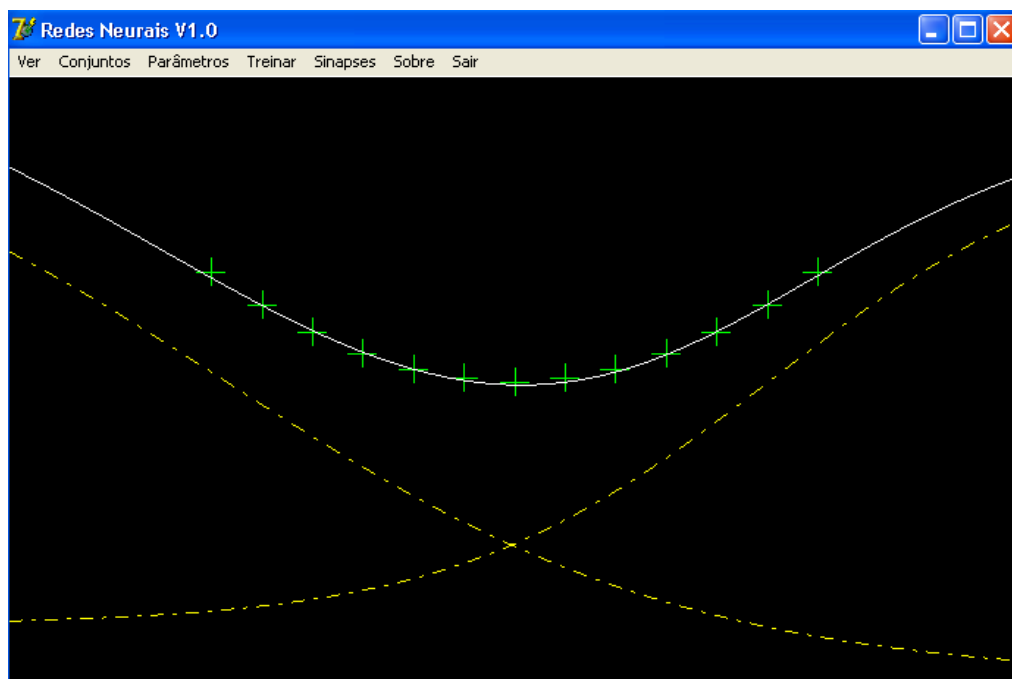
Calôba, Luiz P. – *Notas de Aula. CPE 721. Redes Neurais FeedForward* – COPPE/UFRJ, 2004.

Haykin, Simon – *Redes Neurais – Princípios e Prática* – 2ª ed – Bookman, 2002.

Wasserman, Philip D. – *Neural Computing. Theory and Practice* – Van Nostrand Reinhold, 1989.

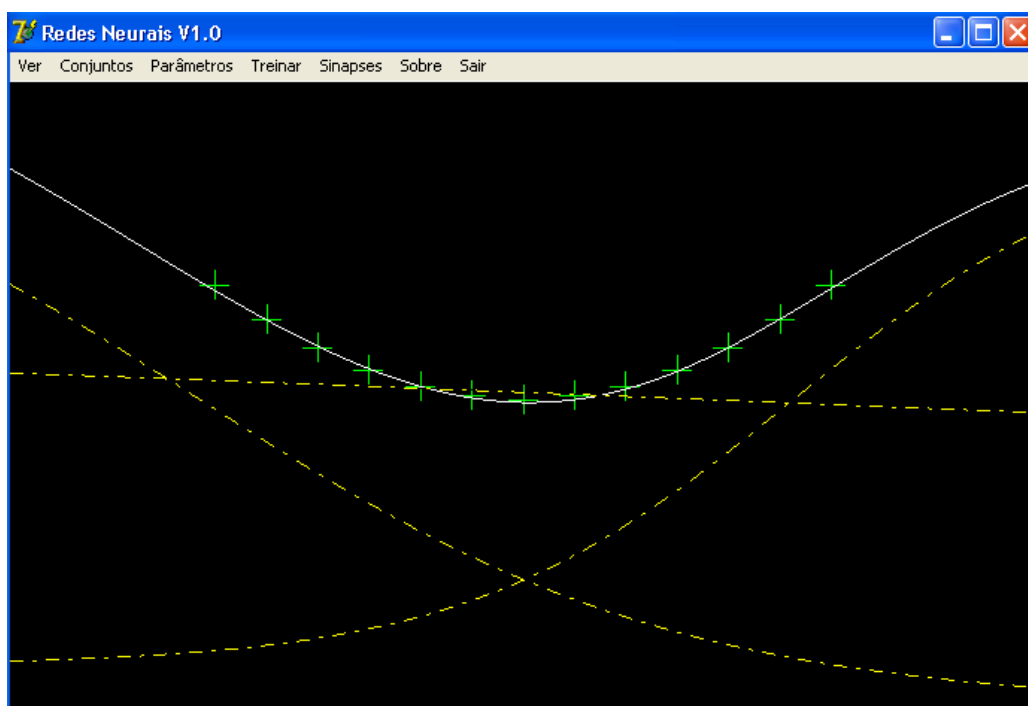
**APÊNDICE A**

**CASOS  
SELECIONADOS**



**Figura A1 – Conjunto de treinamento X2.trn, 2 neurônios na camada oculta, 100000 passos de treinamento, taxa de aprendizagem 10.**

Na figura A1, vemos que a curva em branco (saída da rede) pode ser decomposta nas duas curvas em amarelo (saídas dos neurônios da camada intermediária).

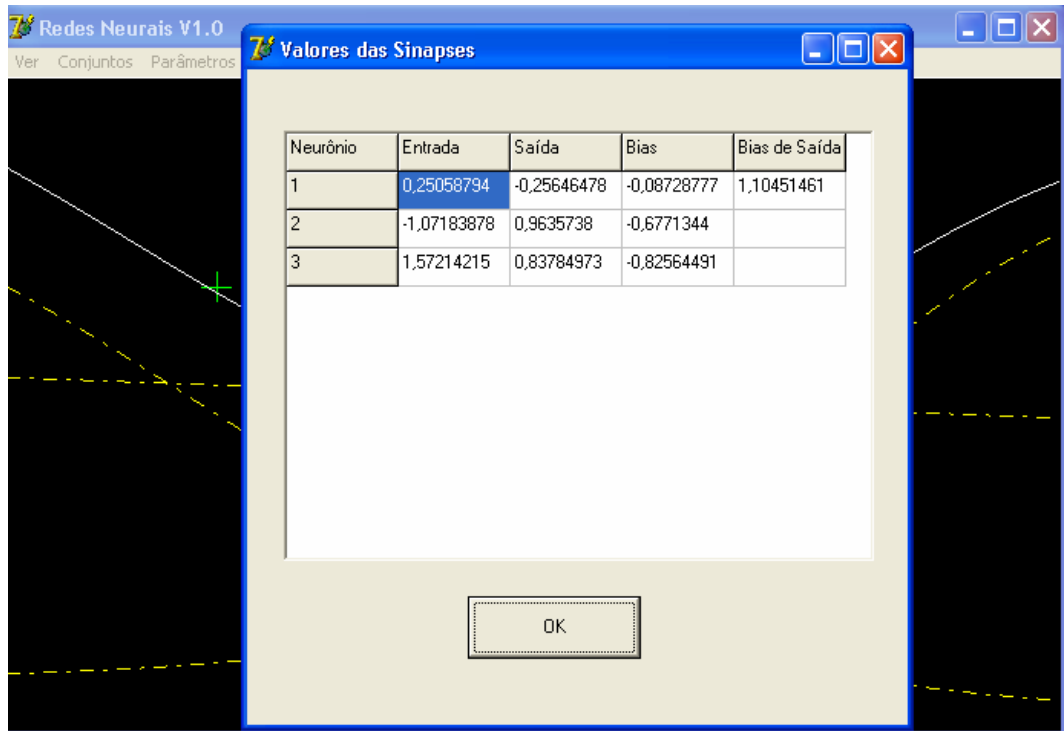


**Figura A2 - Conjunto de treinamento X2.trn, 3 neurônios na camada oculta, 100000 passos de treinamento, taxa de aprendizagem 10.**

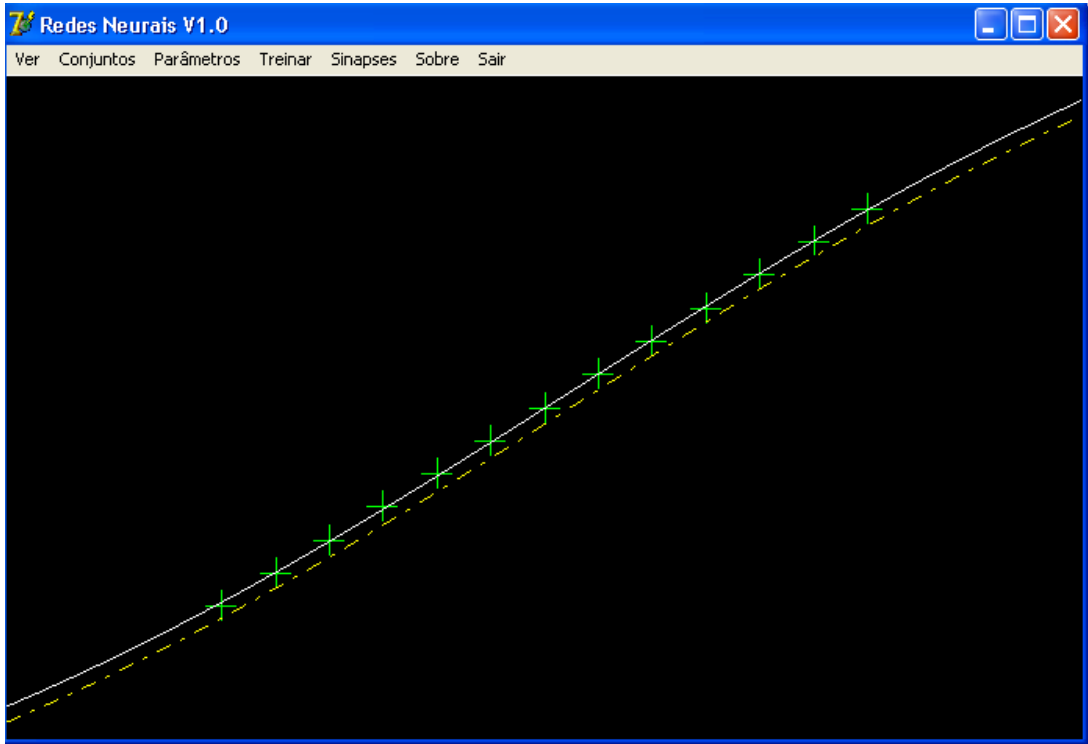
A figura A2 mostra claramente que um dos neurônios está “ocioso”, ou seja, esta função é perfeitamente representada com apenas 2 neurônios. Isto também pode ser



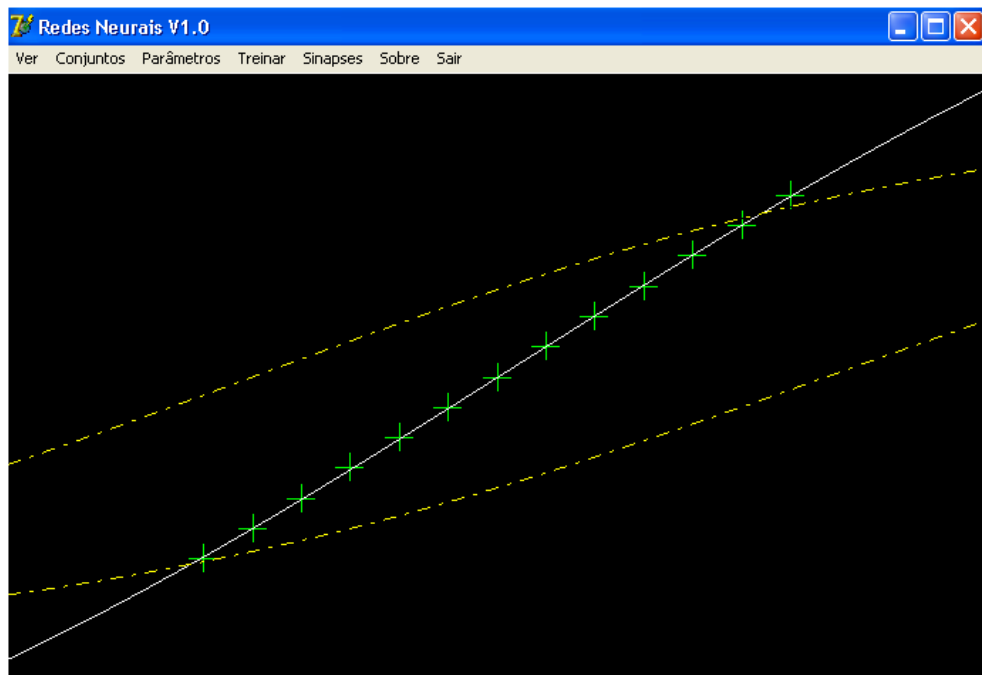
constatado quando abrimos a tabela com os valores das sinapses e observamos o neurônio 1 (figura A3).



**Figura A3 – Valores das sinapses para o caso mostrado na figura A2**



**Figura A4 - Conjunto de treinamento X.trn, 1 neurônio na camada oculta, 100000 passos de treinamento, taxa de aprendizagem 10.**

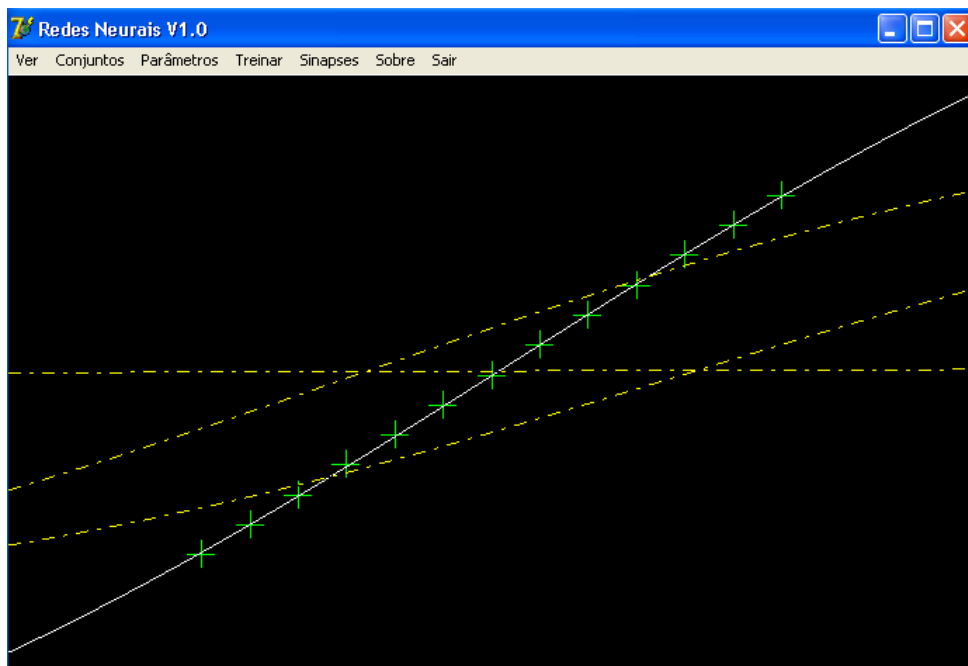


**Figura A5 - Conjunto de treinamento X.trn, 2 neurônios na camada oculta, 100000 passos de treinamento, taxa de aprendizagem 10.**

A figura A4 mostra uma função de primeiro grau sendo mapeada por apenas um neurônio. As linhas não são coincidentes apenas porque o bias do neurônio de saída desloca a curva amarela para a posição da curva branca.

Esperávamos que, com 2 neurônios na camada oculta, um deles ficasse “ocioso”. Não foi o que observamos na figura A5, onde os dois neurônios contribuem de forma significativa com a saída.

Já com 3 neurônios podemos observar novamente um forte candidato ao “pruning” (figura A6).



**Figura A5 - Conjunto de treinamento X.trn, 3 neurônios na camada oculta**

# **APÊNDICE B**

## **TRANSPARÊNCIAS UTILIZADAS**

# **APÊNDICE C**

## **PROGRAMAS FONTE**

# **APÊNDICE D**

## **ALTERAÇÕES NO DOCUMENTO**

Após a entrega do trabalho na data prevista, o programa foi modificado a pedido do professor.

Dentre as alterações citamos:

- 1- A inclusão do conjunto de teste;
- 2- A inclusão dos gráficos dos erros médios quadráticos dos conjuntos de treinamento e teste;
- 3- O parâmetro 'Taxa de Aprendizagem' foi modificado para permitir valores com 3 casas decimais;
- 4- É permitido ao usuário alterar o módulo dos valores para o chute inicial das sinapses.

# **APÊNDICE E**

## **COMO CRIAR OS CONJUNTOS DE TREINAMENTO E TESTE**

Para a criação dos arquivos de treinamento e teste devemos proceder da seguinte maneira:

Abrir o Bloco de Notas do Windows e escrever o primeiro par do conjunto de treinamento, x e y, nesta ordem, separados por espaços. Para separar as casas decimais devemos usar o ponto. Proceda de forma semelhante para entrar com os outros pares do conjunto de treinamento, pressionando a tecla <Enter> ao final de cada par. Para concluir, selecionar o menu '*Arquivo -> Salvar Como ...*', na opção '*Salvar como tipo:*' escolher '*Todos os arquivos*' e na opção '*Nome do arquivo:*', colocar o nome desejado com a extensão '*.trn*' .

O procedimento é igual para o conjunto de teste, sendo que é recomendável utilizar o mesmo nome do arquivo de treinamento, mas com a extensão '*.tst*' .

Obs: Os pares de treinamento e de teste já devem ser fornecidos normalizados !

Obs2: A criação do arquivo de teste é opcional.

E-mails para contato:

[marcos\\_mello@yahoo.com.br](mailto:marcos_mello@yahoo.com.br)

[jlpaz2004@yahoo.com.br](mailto:jlpaz2004@yahoo.com.br)