ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

Departamento de Engenharia de Computação e Sistemas Digitais

Redes Neurais Multicamadas

Seminário de Inteligência Artificial

Autores:	Data de Emissão:
André Kenji Horie *Hugo Brizard* *Ricardo Henrique Gracini Guiraldelli*	*13/11/2008*

Índice

1. Introdução

1.1. Redes Neurais Biológicas

Na neurociência, uma rede neural descreve uma população de neurônios interligados fisicamente. A comunicação entre neurônios frequentemente envolve um processo eletroquímico, e a interface através da qual eles se interagem consiste de vários dendritos (conexões de entrada), que são interligados através de sinapses com outros neurônios, e um axônio (conexão de saída). Se a soma dos sinais de entrada ultrapassa um determinado limiar, o neurônio envia uma ação potencial (AP), e o axônio transmite este sinal elétrico.

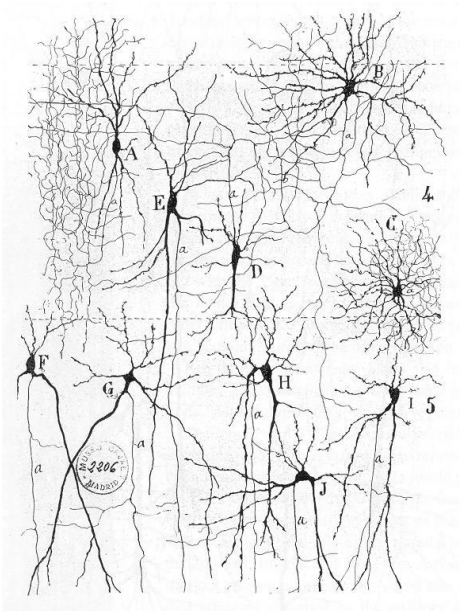

Figura 1.1: Rede neural biológica

1.2. Redes Neurais Artificiais

Uma rede neural artificial (ANN), frequentemente denominada apenas "rede neural" (NN), é um modelo matemático ou computacional baseado em redes neurais biológicas. Consiste de um grupo interligado de neurônios artificiais que processa informação utilizando uma abordagem computacional. Na maioria dos casos, uma ANN é um sistema adaptativo que muda sua estrutura interna ou externa baseada na informação que flui através da rede, durante a fase da aprendizagem.

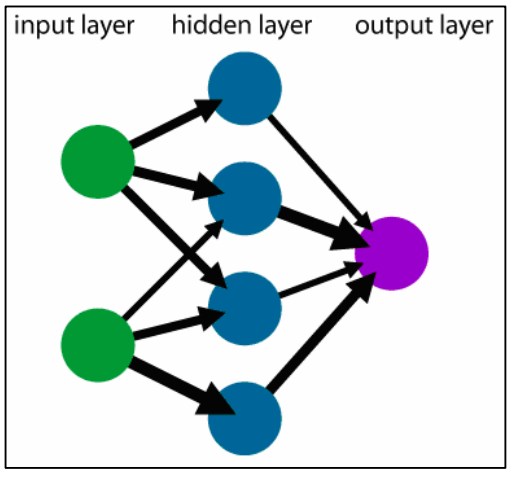

Figura 1.2: Rede neural artificial

Em termos mais práticos, redes neurais são ferramentas não-lineares para modelagem de dados estatísticos. Elas podem ser usadas para modelar as relações complexas entre entradas e saídas, ou para encontrar padrões em dados.

Pode-se afirmar que as redes neurais biológicas e artificiais são semelhantes por várias causas. Primeiro, se um indivíduo coloca a mão no fogo, a dor será quase instantânea. São as reações da rede neural. Semelhante com as redes neurais artificiais, o indivíduo aprenderá que fazer isso não dá um resultado feliz. As redes neurais artificiais podem treinar com o tempo para otimizar

os resultados, para não cometer erros novamente, ou para minimizar ao máximo esses erros.

Segundo, a estrutura das redes são muito parecidas: ambas possuem ligações de entrada e de saída, com cálculos que decidem o resultado. Na verdade, as redes neurais artificiais foram criadas para melhor entender as redes neurais biológicas. Depois, muitas aplicações diferentes foram criadas. Alguns exemplos serão providos posteriormente.

Existem vários tipos de redes neurais artificiais, com suas vantagens e desvantagens, considerando-se os resultados obtidos e o tempo exigido para o processamento.

1.3. Tipos de Problemas

A utilidade do modelo das redes neurais artificiais reside no fato que elas podem ser usadas para inferir uma função a partir de observações. Isto é particularmente útil em aplicações nas quais a complexidade das tarefas ou dos dados a serem processados torna a modelagem de tal função impraticável.

As redes neurais multi-camadas são normalmente utilizadas para tipos de problemas particulares. São eles:

- Classificação: A rede neural é treinada através de amostras de dados de cada grupo e instruções de como realizar a classificação, fazendo com que a rede aprenda as características próprias de cada grupo.
- Função aproximação (ou análise de regressão): Inclui previsão e modelagem de séries temporais.
- Reconhecimento de padrões: É uma forma de classificação. Redes neurais são largamente utilizadas para este intuito.
- Otimização: Problemas como montagem de circuitos, alocação de recursos, entre outros.

Outras áreas de aplicação incluem: sistema de identificação e controle (controle veicular, de processos, entre outros), tomada de decisões em jogos (gamão, xadrez, corridas), diagnósticos médicos, aplicações financeiras (sistemas de negociação automáticos), *Data Mining* (descoberta de conhecimento nas bases de dados, ou KDD), visualização e filtragem de spam de e-mail.

1.4. Exemplos de Aplicações

- Reconhecimento de escrita (http://neurovision.sourceforge.net/).
- Predição de localização (http://sourceforge.net/projects/smith-project/).

2. Redes Neurais Multicamadas

2.1. O que são Multicamadas?

As redes neurais de camada única possibilitam apenas a resolução de problemas linearmente separáveis, i.e., as diversas soluções (booleanas) para os problemas estão separados por um plano. As redes neurais multicamadas, por outro lado, possibilitam a resolução de um espectro maior de problemas através da adição das camadas escondidas, incluindo problemas discretos e regressões não-lineares, aumentando o espaço de hipóteses que a rede pode representar e possibilitando grande poder computacional.

Por exemplo, na figura 2.1a abaixo, temos a saída de um perceptron com uma função de ativação sigmoidal. Combinando-se duas sigmóides voltadas para direções opostas, obtém-se um "cume", ilustrado na figura em 2.1b. Por fim, combinando-se dois cumes, o resultado é a figura 2.1c.

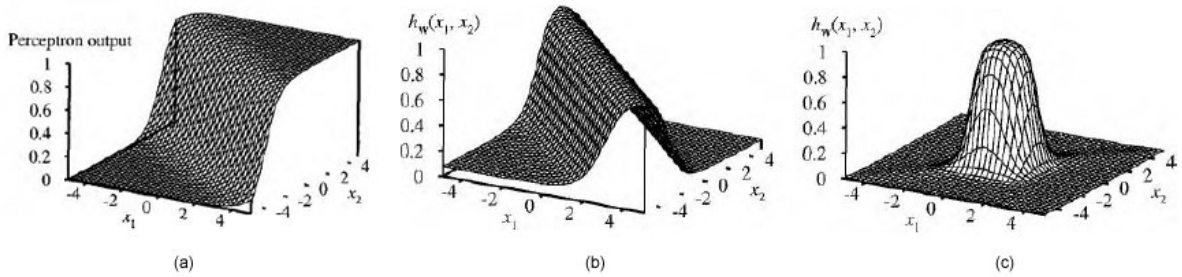

Figura 2.1: Saídas de perceptrons com função de ativação sigmoidal.

Este aumento de poder computacional não se dá apenas ao fato de se aumentar o número de camadas da rede. Dadas as funções de ativação como a função geradora da saída de cada unidade da rede, é possível provar por Álgebra Linear que se esta função de ativação for linear em todos os neurônios, esta rede pode ser reduzida para uma rede neural de uma única camada.

Assim, o que difere a rede multicamadas são suas funções não-lineares, como por exemplo a função de Heaviside ou uma função sigmoidal (figura 2.2). Observa-se preferência para as sigmóides por elas serem normalizáveis e diferenciáveis.

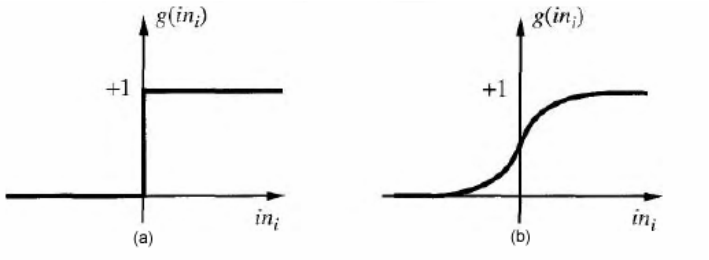

Figura 2.2: (a) Função Heaviside, (b) Função Sigmoidal

De acordo com o Teorema de Cybenko[4], *qualquer função contínua* pode ser uniformemente aproximada por uma rede neural contínua de apenas uma camada interna e com uma sigmoidal contínua não-linear arbitrária. De fato, qualquer função descontínua pode ser aproximada com duas camadas escondidas[1]. Infelizmente, para uma dada estrutura de rede, é muito difícil caracterizar as funções que podem ser representadas.

2.2. Feed-Forward

Redes que podem ser representadas por grafos direcionados acíclicos são chamadas de feed-forward. Em outras palavras, as saídas de todas as unidades do perceptron vão para as camadas posteriores, não havendo assim loops de feedback. As redes feed-forward representa uma função da entrada atual e, portanto, não tem um estado interno além dos pesos pré-definidos.

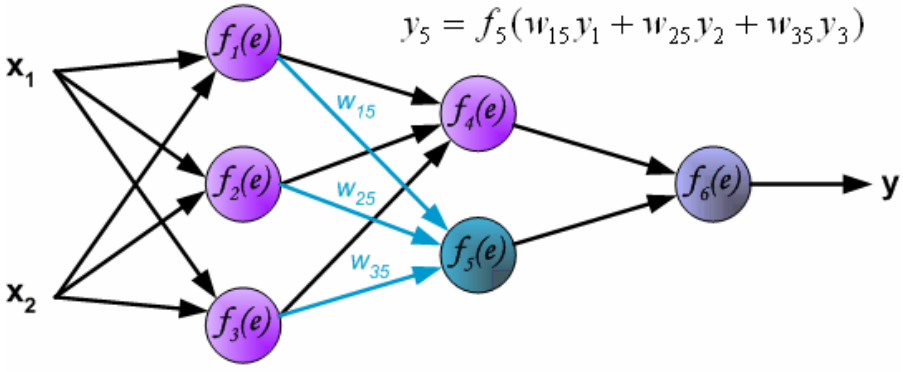

Figura 2.3: Exemplo gráfico de Feed-Forward.

2.3. Back-Propagation

Sabe-se que para as redes neurais de camada única, o aprendizado é realizado facilmente devido à simplicidade no cálculo do erro de comparação entre a saída esperada e a saída da rede. Contudo, nas redes multicamadas não é possível calcular diretamente o erro para as unidades escondidas, uma vez que não são definidas as

saídas destas. Assim, sem uma técnica de aprendizado, redes neurais multicamadas tornam-se inúteis.

Para solucionar esse problema, em 1986, foi proposto por Rumelhart, Hinton e Williams[6] o algoritmo (ou método) de back-propagation. Este algoritmo tem como idéia principal propagar o erro da saída do perceptron de volta para as camadas anteriores, de forma que todas as unidades da rede possam aprender com o erro.

A definição matemática, algoritmo e um exemplo serão apresentados no capítulo seguinte.

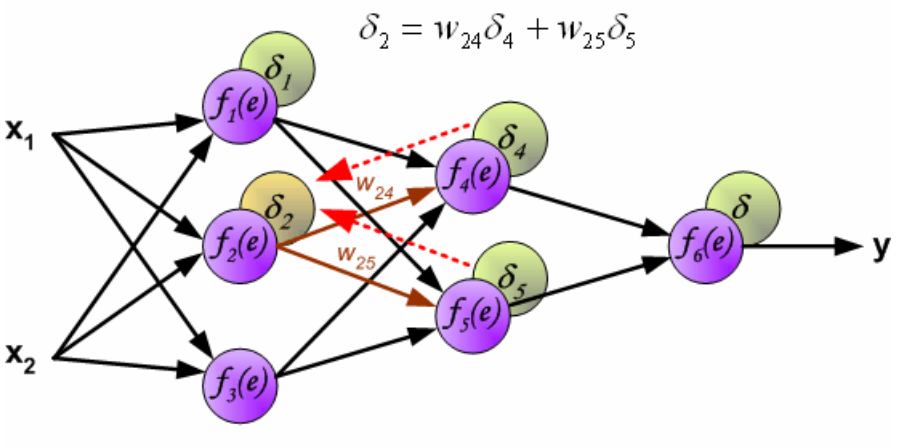

Figura 2.4: Exemplo gráfico de Feed-Forward com Back-Propagation.

2.4. Redes Neurais Recorrentes

As redes neurais recorrentes (RNN) formam uma classe de redes neurais na qual as conexões entre as unidades formam um grafo direcionado cíclico, criando assim um estado interno que exibe comportamento temporal dinâmico. Este comportamento pode ser tanto estável, como oscilante e até mesmo caótico.

Uma das principais motivações para as RNN's provem do fato que as chamadas RNN's analógicas (ARNN) são equivalentes às máquinas de Turing[5]. Mais que isso: enquanto muitos sistemas dinâmicos e caóticos não podem ser descritos por uma máquina de Turing, eles podem ser capturados por ARNN's, suportando assim a proposição que estas descrevem a computação analógica da mesma forma que as TM's descrevem a computação digital [7] [8] [9]. Esta expansão do modelo de Turing pode ser vista também como uma mudança da computação estática para a adaptativa, sendo denominada de poder de computação de super-Turing ou hipercomputação.

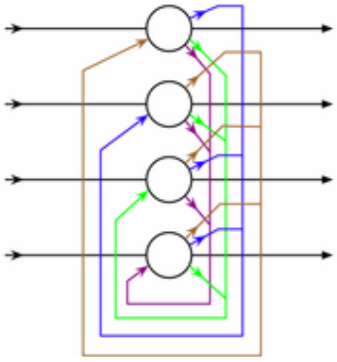

Figura 2.5: RNN de Hopfield

3. Back-Propagation

3.1. Descrição Matemática

Para cada elemento (perceptron ou neurônio da rede), definimos sua soma ponderada como $s_i = \sum_{j=0}^{N} w_{ij} x_j$, onde x_j é j-ésima entrada no neurônio i e w_{ij} é o peso da respectiva entrada neste neurônio.

A saída do neurônio, yi, é definida por uma função contínua diferenciável de si, σi: $y_i = \sigma_i(s_i)$, onde yi assume um dos seguintes valores:

Função logística: $y_i = \frac{1}{1 + e^{-s_i}}$ ou

Função tangente hiperbólica: $y_i = \tanh(s_i)$.

Vamos, para fins de dedução, estudar uma rede de três camadas, sendo uma camada de entrada, uma camada escondida (*hidden layer*) e uma camada de saída, conforme figura a seguir:

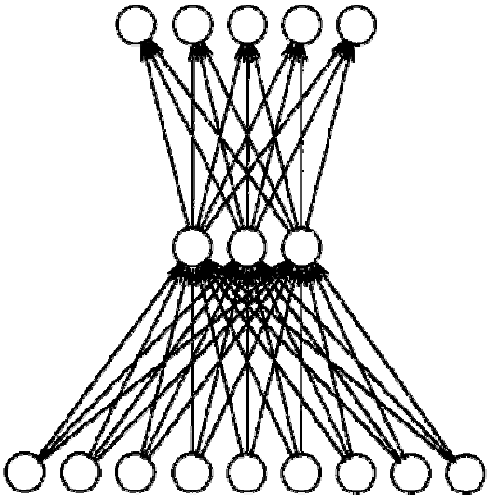

Figura 3.1: Rede neural de três camadas.

Definindo cada uma das saídas da camada escondida como $y_j = \sigma_j^y(\sum_{i=0}^{I} w_{ji}^y x_i)$ e cada saída da camada de saída como $z_k = \sigma_k^z(\sum_{j=0}^{J} w_{kj}^z y_j)$, procuremos minimizar o erro quadrático da saída da rede.

Dado que o erro quadrático da rede (camada de saída) é $E_p = \sum_{k=1}^{K} (z_{pk} - t_{pk})^2$, sendo t_{pk} a saída "ideal" (valor alvo) e p determinado padrão, calculamos a derivada de E_p (a respeito de um determinado w) para tentarmos minimizar o erro:

$$\frac{dE_p}{dw} = 2\sum_{k=1}^{K} (z_{pk} - t_{pk}) \frac{dz_{pk}}{dw}$$

Se $w = w_{pj}^z$ é um dos pesos da camada de saída:

$$\frac{dE_p}{dw_{kj}^z} = 2(z_{pk} - t_{pk}) \times \sigma_{pk}^{z'} \times y_{pj}$$

Se $w = w_{pi}^y$ é um dos pesos da camada escondida (hidden layer):

$$\frac{dE_p}{dw_{ji}^y} = 2\sum_{k=1}^{K} (z_{pk} - t_{pk}) \times \sigma_{pk}^{z'} \times w_{kj}^z \times \sigma_{pj}^{y'} \times x_{pi}$$

Dado esses resultados, definimos $\Delta w = -\eta \times \nabla E_p$, onde η é a taxa de apredendizado, que deve ser somado ao peso para ajuste de seu aprendizado ($w_j^n = w_j^{n-1} + \Delta w_j$).

3.2. Algoritmo

1- Defina o número de *camadas ocultas* e o número de neurônios em cada camada.

2- Defina valores iniciais, possivelmente randômicos, para os pesos.

3- Escolha um par de padrões (x_p, t_p) e faça as condições iniciais x_p.

4- Calcule as somas ponderadas e as saídas dos neurônios, *y*, propagando os valores para as camadas "da frente".

5- Calcule os erros entre o valor das saídas *reais* e *alvo/objetivo*.

6- Faça a minimização dos erros, calculando as derivadas.

7- Calcule os valores de Δw para os pesos de cada camada.

8- Faça $w_j^n = w_j^{n-1} + \Delta w_j$ para cada peso.

9- Repita os passos de 3-8 para todo o corpo de treinamento.

10- Repita os passos 3-9 até que a rede tenha um resultado satisfatório.

3.3. Exemplo

Dada a função de ativação logística $y_i = \frac{1}{1+e^{-s_i}}$, o perceptron ao lado, e os seguintes padrões a serem aprendidos:

- Para entradas 0 e 1, a saída deve ser 0;
- Para entradas 1 e 1, a saída deve ser 1.

A taxa de aprendizado da rede será de 0,25.

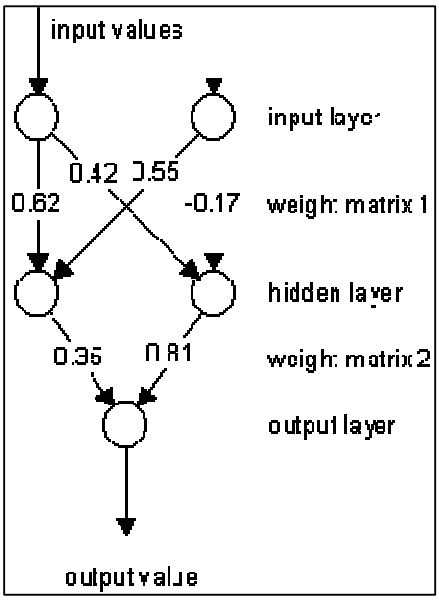

Figura 3.2: Perceptron do exemplo

Inicialmente, devemos aplicar o Feed-Forward através da soma das entradas levando em consideração os pesos apresentados e a função de ativação. Desta forma, para a camada escondida, teremos:

- Entrada do neurônio escondido 1:

$$s_i = \sum_{j=0}^{N} wi_j x_j = 0 \times 0{,}62 + 1 \times 0{,}55 = 0{,}55$$

- Entrada do neurônio escondido 2:

$$s_i = \sum_{j=0}^{N} wi_j x_j = 0 \times 0{,}42 + 1 \times (-0{,}17) = -0{,}17$$

- Saída do neurônio escondido 1:

$$y_i = \frac{1}{1+e^{-s_i}} = \frac{1}{1+e^{-0{,}55}} = 0{,}634135591$$

- Saída do neurônio escondido 2:

$$y_i = \frac{1}{1+e^{-s_i}} = \frac{1}{1+e^{0{,}17}} = 0{,}457602059$$

Desta forma, a saída do sistema obtida será:

$$s_i = \sum_{j=0}^{N} wi_j x_j = 0{,}634135591 \times 0{,}35 + 0.457602059 \times 0{,}81 = 0{,}592605124$$

$$y_i = \frac{1}{1+e^{-s_i}} = \frac{1}{1+e^{-0{,}592605124}} = 0{,}643962658$$

$$\therefore \quad Erro = E = 0 - 0{,}643962658 = -0{,}643962658$$

Agora, iniciaremos o Back-Propagation. Para este caso, o novo valor de cada peso será dado por $w_{ji} \leftarrow w_{ji} + \eta \times z \times E \times y_j \times y_i$, onde w_{ji} é o peso, η é a taxa de aprendizado, z é a saída da rede, E é o erro na saída da rede, y_j é a saída do nó que origina a transição referente ao peso e y_i é a saída do nó no qual a transição termina.

Considerando a matriz de pesos número 2 (entre a camada escondida e a camada de saída), temos respectivamente:

(i) $w_{ji} \leftarrow w_{ji} + \eta \times z \times E \times y_j \times y_i$

$$w_{ji} = 0{,}35 + \left[0.25 \times (-0{,}643962658) \times 0{,}634135591 \times 0{,}643962658 \times (1 - 0{,}643962658)\right]$$

$$w_{ji} = 0{,}35 + (-0{,}023406638) = 0{,}326593362$$

(ii) $w_{ji} \leftarrow w_{ji} + \eta \times z \times E \times y_j \times y_i$

$$w_{ji} = 0{,}35 + \left[0.25 \times (-0{,}643962658) \times 0{,}457602059 \times 0{,}643962658 \times (1 - 0{,}643962658)\right]$$

$$w_{ji} = 0{,}35 + (-0{,}016890593) = 0{,}793109407$$

Por fim, para a matriz de pesos número 1 (entre a entrada e a camada escondida), temos:

(i) $w_{ji} \leftarrow w_{ji} + \eta \times z \times E \times y_j \times y_i$

$$w_{ji} = 0{,}62 + \left[0.25 \times (-0{,}643962658) \times 0 \times 0{,}634135591 \times (1 - 0{,}634135591)\right] = 0{,}62$$

(ii) $w_{ji} \leftarrow w_{ji} + \eta \times z \times E \times y_j \times y_i$

$$w_{ji} = 0{,}42 + \left[0.25 \times (-0{,}643962658) \times 0 \times 0{,}457602059 \times (1 - 0{,}457602059)\right] = 0{,}42$$

(iii) $w_{ji} \leftarrow w_{ji} + \eta \times z \times E \times y_j \times y_i$

$$w_{ji} = 0{,}55 + \left[0.25 \times (-0{,}643962658) \times 0 \times 0{,}634135591 \times (1 - 0{,}634135591)\right]$$

$$w_{ji} = 0{,}55 + (-0{,}037351064) = 0{,}512648936$$

(iv) $w_{ji} \leftarrow w_{ji} + \eta \times z \times E \times y_j \times y_i$

$$w_{ji} = -0{,}17 + \left[0.25 \times (-0{,}643962658) \times 0 \times 0{,}457602059 \times (1 - 0{,}457602059)\right]$$

$$w_{ji} = -0{,}17 + (-0{,}039958271) = -0.209958271$$

O mesmo procedimento deve ser aplicado para os próximos valores de aprendizado, mas agora utilizando os pesos novos.

4. Conclusão

As redes neurais possibilitam multicamadas apresentam uma evolução no poder computacional em relação às redes de camada única devido à sua maior capacidade de representação, esta explicada pelo uso de camadas escondidas e de funções de ativação não-lineares. Embora nas multicamadas o erro não seja tão evidente, ele é possível de ser calculado através de técnicas como o Back-Propagation, permitindo assim o aprendizado por treinamento, principal característica das redes neurais.

Observou-se as diversas aplicações nas quais é possível utilizar esta técnica de Inteligência Artificial e também a analogia entre as RNN's e a máquina de Turing.

Referências

[1] RUSSEL, S. J.; NORVIG, P. *Artificial Intelligence*: A Modern Approach. 2ª edição. New Jersey: Prentice Hall, 2002. 1132 p.

[2] http://neurovision.sourceforge.net/ (acessado em 10/11/2008).

[3] http://sourceforge.net/projects/smith-project/ (acessado em 10/11/2008).

[4] CYBENKO, G. V. *Approximation by Superpositions of a Sigmoidal function*. Mathematics of Control, Signals and Systems, vol. 2 no. 4 pp. 303-314. 1989.

[5] HYOTYNIEMI, H. *Turing Machines are Recurrent Neural Networks*. 1996.

[6] RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning Internal Representations by Error Propagation*. In Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations, D. E. Rumelhart and J. L. McClelland, Eds. Mit Press Computational Models Of Cognition And Perception Series. MIT Press, Cambridge, MA, 318-362. 1986.

[7] SIEGELMANN, H. T. *Neural and Super-Turing Computing*. Minds Mach. 13, 1, 103-114. 2003.

[8] SIEGELMANN, H. T. *The Simple Dynamics of Super-Turing Theories*. Theor. Comput. Sci. 168, 2, 461-472. 1996.

[9] http://binds.cs.umass.edu/anna_cp.html (acessado em 10/11/2008).

[10] BODEN, M. A. *Artificial Intelligence*. 1st. Morgan Kaufmann Publishers Inc. 1996.

[11] NILSSON, N. J. 1998 *Artificial Intelligence: a New Synthesis*. Morgan Kaufmann Publishers Inc. 1998.