ESTRUTURA E APRENDIZADO DE UMA REDE NEURAL NEBULOSA RECORRENTE

R. Ballini[1]; S. Soares[2]; F. Gomide[1]

[1]*Departamento de Engenharia de Computação e Automação Industrial*
Fac. de Eng. Elétrica e de Computação, Universidade Estadual de Campinas
CP 6101, 13083-970, Campinas - SP - Brasil
E-mails: {ballini, gomide}@dca.fee.unicamp.br

[2]*Departamento de Engenharia de Sistemas*
Fac. de Eng. Elétrica e de Computação, Universidade Estadual de Campinas
CP 6101, 13083-970, Campinas - SP - Brasil
E-mail: dino@densis.fee.unicamp.br

Resumo— Um modelo de rede neural nebulosa recorrente é proposto neste artigo. Este modelo é constituído de neurônios lógicos formados a partir de conjuntos nebulosos. A rede tem uma estrutura multi-camadas e recorrente cujas unidades são modeladas através de normas e co-normas triangulares, e pesos definidos dentro de um intervalo unitário. O procedimento de aprendizagem desenvolvido é baseado em dois paradigmas principais: método do gradiente e aprendizado por reforço associativo, respectivamente. Isto é, os pesos da camada de saída são ajustados via um método do gradiente do erro, enquanto que um esquema de recompensa ou punição atualiza os pesos da camada intermediária. A rede neural nebulosa recorrente é usada para obter um modelo de um processo não-linear. Resultados numéricos mostram que a rede neural nebulosa proposta aqui fornece um modelo preciso depois de um curto período de treinamento.

Abstract— A novel recurrent neurofuzzy network is proposed in this paper. This model is constructed from fuzzy set models of neurons. The network has a multilayer, recurrent structure whose units are modeled through triangular norms and co-norms, and weights defined within the unit interval. The learning procedure developed is based on two main paradigms: gradient search and associative reiforcement learning, respectively. That is, output layer weights are adjusted via an error gradient method whereas a reward and punishment scheme updates the hidden layer weights. The recurrent neurofuzzy network is used to develop a model of a nonlinear process. Numerical results show that the neurofuzzy network proposed here provides an accurate process model after a short period of learning time.

Key Words— Recurrent neurofuzzy networks, fuzzy sets, neural networks, function approximation.

1 Introdução

Redes neurais não-recorrentes têm sido sucessivamente aplicadas em modelagem, identificação e controle (Narendra e Parthasarathy, 1990) e em outras áreas. Estes modelos codificam mapeamentos estáticos de entradas/saídas, aproximando funções contínuas com um arbitrário grau de precisão. Recentemente, devido a sua natureza dinâmica, redes neurais recorrentes mostram-se particularmente apropriadas para tratar certos problemas tais como modelagem de sistemas dinâmicos, controle adaptativo, processamento de séries temporais, previsão, e reconhecimento de voz para nomear uns poucos (Ku e Lee, 1995). Sistemas de redes neurais recorrentes aprendem e memorizam informações implícitamente em sua estrutura e pesos. Apesar do considerável potencial e capacidade as redes recorrentes ainda apresentam certas dificuldades de treinamento. Muitos algoritmos de aprendizagem correntemente conhecidos são complexos e lentos (Lee e Teng, 2000).

Como é amplamente conhecido, sistemas baseados em conjuntos nebulosos e redes neurais têm como objetivo explorar a capacidade de processamento do conhecimento humano. Combinações destas duas abordagens, sistemas neurais nebulosos, têm tido sucesso em muitas aplicações. A abordagem neural nebulosa une a teoria de conjuntos nebulosos e redes neurais em um sistema integrado para combinar os benefícios de ambos. Entretanto, uma limitação de muitas redes neurais nebulosas são suas restritas aplicações em domínio e problemas de mapeamentos dinâmicos devido ou à sua estrutura não-recorrente, ou a falhas de eficientes procedimentos de aprendizado para conexões de realimentação. Neste trabalho, propoêm-se uma rede neural nebulosa recorrente com aprendizado supervisionado. A rede estabelece um mapeamento dinâmico, sendo mais maleável para modelagem de sistemas dinâmicos que redes neurais nebulosas clássicas. De particular interesse, é sua capacidade de tratar com entradas e saídas variando no tempo através de sua operação temporal. Esta habilidade em armazenar informações temporais induz uma estrutura de rede mais simples e eficiente que as redes estáticas correspondentes.

O modelo proposto é uma extensão da rede neural nebulosa não-recorrente introduzida em (Caminhas et al., 1999) e introduz uma estrutura geral e domínio de aplicações mais abrangentes. A estrutura da rede é composta de unidades de

processamento nebulosas modeladas por neurônios lógicos do tipo *and* e *or* (Pedrycz e Rocha, 1998). A rede codifica implícitamente um conjunto de regras se-então em sua estrutura, formando uma relação dual entre a rede neural e o sistema nebuloso associado. Portanto, o conhecimento pode ser facilmente inserido ou extraído da estrutura do modelo.

A rede neural nebulosa recorrente (RNNR) é particularmente apropriada para modelar sistemas dinâmicos não-lineares (Ballini, Soares e Gomide, 2001). Sua estrutura recorrente multi-camadas realiza implícitamente inferência nebulosa. As relações temporais são estabelecidas na segunda camada da rede. Esta modificação fornece os elementos de memória e expande a capacidade básica da rede neural nebulosa recorrente para incluir representações temporais. Como um neurônio recorrente tem uma realimentação interna, este modelo de neurônio captura a resposta dinâmica de um sistema, podendo ser simplificado. O algoritmo de aprendizagem é baseado no gradiente da função erro e no aprendizado associativo por reforço originado em (Barto e Jordan, 1987). A rede neural nebulosa recorrente foi usada para modelar um processo não-linear. Resultados numéricos mostram que a rede neural nebulosa proposta aqui fornece um modelo preciso, com ordem de grandeza nos erros de teste na ordem de 10^{-4}, depois de um pequeno período de treinamento.

2 Estrutura da Rede Neural Nebulosa Recorrente

As unidades de processamento básicas consideradas aqui são dois neurônios nebulosos chamados neurônios lógicos *and* e *or*, produzindo mapeamentos $[0,1]^n \rightarrow [0,1]$ (Pedrycz e Rocha, 1998), (Pedrycz e Gomide, 1998). A implementação padrão de conectivos de conjuntos nebulosos envolvem normas triangulares, significando que os operadores *and* e *or* são implementados através das t- e s- normas, fornecendo uma estrutura de neurônio na qual $w_i \in [0,1]$ é o peso associado com a entrada $x_i \in [0,1]$ (Figura 1).

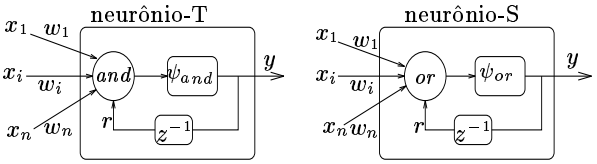

Figura 1. Neurônios lógicos recorrentes.

A rede neural tem uma estrutura multi-camadas com realimentação na saída dos neurônios *and* da segunda camada (Figura 2). A camada de entrada consiste de neurônios cujas funções de ativação são funções de pertinência dos conjuntos nebulosos que formam a partição do espaço de entrada. Ou seja, para cada dimensão $x_i(t)$ de um vetor de entrada n-dimensional $\mathbf{x}(t)$ existem N_i conjuntos nebulosos $A_i^{k_i}$, $k_i = 1, \ldots, N_i$ cujas funções de pertinência são funções de ativação dos correspondentes neurônios de entrada. A variável t denota o tempo discretizado, isto é, $t = 1, 2, \ldots$, e será omitida no decorrer do artigo para simplificar a notação. Assim, as saídas desta camada são graus de pertinência associados aos valores de entrada, isto é, $a_{ji} = \mu_{A_i^{k_i}}(x_i)$, $i = 1, \ldots, n$ e $j = 1, \ldots, M$, onde j representa o número de neurônios da segunda camada. Os neurônios da segunda camada são neurônios *and* com realimentação na saída cujas entradas a_{ji} são ponderadas por w_{ji} e as conexões de realimentação são ponderadas por r_j. A função de ativação ψ_{and} é, em geral, um mapeamento não-linear, mas aqui é considerada como sendo a função identidade $\psi(u) = u$.

Também, sem perda de generalidade, foi considerado um simples neurônio *or* não-recorrente na camada de saída. A saída deste neurônio é denotada por y, as entradas por y_j e os pesos por v_j. A função de ativação ψ_{or} do neurônio *or* são mapeamentos não-lineares.

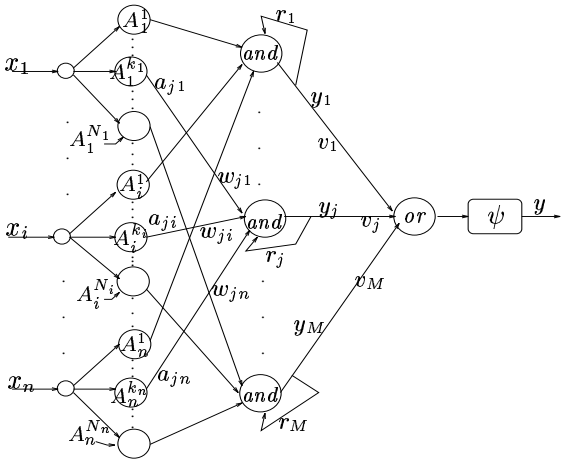

Figura 2. Estrutura da rede neural nebulosa recorrente.

Fazendo $a_{jn+1} = y_j^{-1} = z^{-1} y_j$ (z^{-1} é o operador atraso) e $w_{jn+1} = r_j$, a estrutura da rede envolve um conjunto de regras se-então $\mathrm{R} = \{\mathrm{R}_j, j = 1, \ldots, M\}$ da seguinte forma:

R_j : Se (x_1 é $A_1^{k_1}$ com grau de certeza w_{j1}) ...
E (x_i é $A_i^{k_i}$ com grau de certeza w_{ji}) ...
E (x_n é $A_n^{k_n}$ com grau de certeza w_{jn})
Então z é y_j com grau de certeza v_j

onde $y_j = \mathop{\mathbf{T}}_{i=1}^{n+1} (w_{ji} \, \mathbf{s} \, a_{ji})$. Portanto, existe uma dualidade entre a rede e o conjunto de regras. O esquema de processamento direto inferido na estrutura da rede segue os princípios da teoria de conjuntos nebulosos e raciocínio aproximado.

A estrutura da rede dinâmica pode ser resumida pelos seguintes elementos:

1. N_i é o número de conjuntos nebulosos que constitui a partição da i-ésima entrada;
2. j é o índice que indexa um neurônio *and*. Da estrutura da rede, j é determinado a partir dos índices de entrada k_i como segue:

$$j = k_n + \sum_{i=2}^{n} (k_{(n-i+1)} - 1) \left(\prod_{\tau=1}^{i-1} N_{(n+1-\tau)} \right) \quad (1)$$

onde $K = (k_1 \ \ldots \ k_i \ \ldots \ k_n)$. Para exemplificar, suponha que x_1 é A_1^1; x_2 é A_2^3; x_3 é A_3^2 e $N_1 = N_2 = N_3 = 3$. Neste caso, $k_1 = 1$; $k_2 = 3$ e $k_3 = 2$ e $j = 2 + (k_2 - 1).N_3 + (k_1 - 1).N_3.N_2 = 2 + (3 - 1).3 + (1 - 1).9 = 8$.

3. $x_1, \ldots, x_i, \ldots, x_n$ são n entradas da rede para um dado padrão $\mathbf{x}$;

4. $a_{ji} = \mu_{A_i^{k_i}}(x_i)$ é o grau de pertinência de x_i no conjunto nebuloso $A_i^{k_i}$, sendo uma entrada para o j-ésimo neurônio da segunda camada;

5. w_{ji} é o peso entre a i-ésima entrada e o j-ésimo neurônio *and*;

6. y_j é a saída do j-ésimo neurônio *and*, sendo determinado por:

$$y_j = \mathop{\mathbf{T}}_{i=1}^{n+1} (w_{ji} \, \mathbf{s} \, a_{ji}) \quad (2)$$

7. v_j é o peso para a j-ésima entrada do neurônio *or* e r_j é o peso da conexão recorrente do neurônio j;

8. y é a saída do neurônio *or* dada por:

$$y = \psi \left(\mathop{\mathbf{S}}_{j=1}^{M} (v_j \, \mathbf{t} \, y_j) \right) = \psi(u) \quad (3)$$

onde $\psi : [0,1] \rightarrow [0,1]$ é um mapeamento monotônico não-linear. Neste trabalho, foi considerado ψ como sendo a função logística $1/(1+exp(-u))$.

3 Algoritmo de Aprendizado

O esquema de aprendizado para a rede neural nebulosa recorrente é primeiro apresentado de uma forma geral. Os passos essenciais para implementação serão detalhados a seguir. O algoritmo é uma forma de aprendizado supervisionado.

Algoritmo de Aprendizado da Rede Neural Nebulosa Recorrente:

1. Gere as funções de pertinência;
2. Inicialize os pesos w_{ji}, v_j e r_j;
3. Até que a condição do critério de parada não seja satisfeita faça:

3.1 Apresente um padrão $\mathbf{x}$ à rede, normalmente escolhido aleatoriamente;

3.2 Efetue a fuzzificação;

3.3 Determine os neurônios *and* ativos;

3.4 Atualize os pesos w_{ji}, v_j e r_j;

O algoritmo termina quando ou um nível específico de desempenho é satisfeito ou um número máximo de iterações é alcançado.

3.1 Geração das Funções de Pertinência

Para gerar as funções de pertinência diretamente, foram assumidas funções triangulares, definidas pelos parâmetros $[x_{i_{min}}, \ x_{i_{max}}]$ e c_{ir}.

A forma mais simples constituida por partições complementares e uniformes, pode não ser a mais adequada se um número de padrões está concentrado em certas regiões do espaço de padrões e dispersas em outras. Nestes casos partições não uniformes são mais apropriadas (Figura 3).

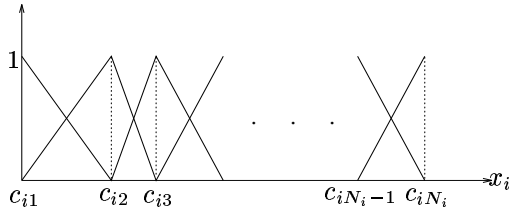

Figura 3. Partições não uniformes.

Técnicas de *clusterização* para encontrar os centros das funções c_{ir} são utilizadas. Em (Caminhas et al., 1999) uma variação de uma rede neural auto-organizada com algoritmo de treinamento tipo LVQ (*Learning Vector Quantization*) (Figura 4) é sugerida.

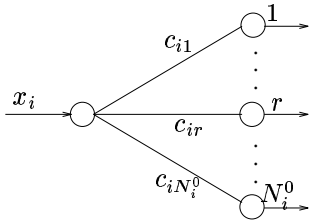

Figura 4. Rede auto-organizada para determinar os centros das funções de pertinência.

Os pesos da rede são os centros c_{ir}. O número de unidades de saída é N_i^0, um dado parâmetro que fornece uma estimação do número de conjuntos nebulosos da i-ésima partição. Em geral, o valor deste parâmetro é sobre-estimado e corrigido via aprendizagem. Neurônios que apresentam um baixo índice de desempenho, isto é, vencem poucas vezes, são removidos. O algoritmo é como segue.

Algoritmo para Gerar as Funções de Pertinência:

1. Inicializações

1.1 Pesos c_{ir}:

$$c_{i1} = x_{i_{min}} \quad (4)$$

$$c_{ir} = c_{i(r-1)} + \Delta_i, \text{ para } r = 2, \ \ldots, \ N_i^0 \quad (5)$$

onde

$$\Delta_i = \frac{x_{i_{max}} - x_{i_{min}}}{N_i^0 - 1} \quad (6)$$

1.2 Índice de desempenho: $id_i(r) = 0$, para $r = 1, \ \ldots, \ N_i^0$.

As funções de pertinência são inicializadas formando partições uniforme. Este procedimento de inicialização geralmente fornece uma convergência mais rápida que a inicialização aleatória.

2. Até que $|c_{ij}(t+1) - c_{ij}(t)| \leq \epsilon$, para$\forall j$ faça:

2.1 Apresente um padrão de entrada x_i e atualize o peso do neurônio vencedor como segue:

$$c_{iL}(t+1) = c_{iL}(t) + \alpha(t).[x_i - c_{iL}(t)] \quad (7)$$

onde L é o índice do neurônio vencedor, para o qual:

$$L = \arg\left\{\min_r |x_i - c_{ir}|\right\} \quad (8)$$

2.2 Reduza o tamanho do passo $\alpha(t)$;

2.3 Atualize o índice de desempenho do neurônio vencedor, ou seja,

$$id_i(L) = id_i(L) + 1 \quad (9)$$

Elimine todos os neurônios para o qual $id_i \leq \delta$, onde δ é um inteiro limitante positivo. Seja Nne_i o número de neurônios removidos para a i-ésima partição. Assim, o número de conjuntos nebulosos da i-ésima partição é:

$$N_i = N_i^0 - Nne_i \quad (10)$$

Uma vez definidos o número de funções de pertinência N_i e os respectivos centros c_{ir}, o cálculo das funções de pertinência é dado da seguinte forma:

$$\mu_{A_i^r}(x_i) = \begin{cases} \alpha_{eir}.(x_i - c_{ir}) + 1, & c_{ir-1} \leq x_i \leq c_{ir} \\ \alpha_{dir}.(x_i - c_{ir}) + 1, & c_{ir} < x_i \leq c_{ir+1} \\ 0, & \text{caso contrário} \end{cases} \quad (11)$$

onde $\alpha_{eir} = \frac{1}{c_{ir} - c_{ir-1}}$, $\alpha_{dir} = \frac{1}{c_{ir} - c_{ir+1}}$, $i = 1, \ldots, n$ e $r = 1, \ldots, N_i$.

3.2 Inicialização dos Pesos

Os neurônios da primeira e da segunda camada estão inteiramente conectados. A inicialização dos pesos w_{ji}, v_j e r_j é da seguinte forma:

$$w_{ji} = 0\ ;\ \forall i = 1, \ldots, n \text{ e } j = 1, \ldots, M \quad (12)$$

$$v_j = 1.0\ ;\ \forall j = 1, \ldots, M \quad (13)$$

$$r_j = 0\ ;\ \forall j = 1, \ldots, M \quad (14)$$

3.3 Determinação dos Neurônios Ativos

Para cada padrão de entrada, existem pelo menos dois graus de pertinência diferente de zero para cada uma das coordenadas (Figura 3). As funções de pertinência correspondentes definem os neurônios *and* ativos, sendo facilmente identificados como segue.

Para cada padrão de entrada $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_n)$, seja $K^1 = (k_1^1 \ldots k_i^1 \ldots k_n^1)$ um vetor cujos componentes são os índices da primeira funcão de pertinência para cada partição i para o qual o grau de pertinência é não nulo. Seja $K^2 = (k_1^2 \ldots k_i^2 \ldots k_n^2)$ um vetor tal que

$$k_i^2 = \begin{cases} k_i^1 + 1, \text{ se } \mu_{A_i^{k_i^1}}(x_i) \neq 1 \\ k_i^1, \text{ caso contrário} \end{cases} \quad (15)$$

O número de neurônios *and* ativos é $Na = 2^{Pa} \leq 2^n$ onde Pa é o número de elementos tal que $k_i^1 \neq k_i^2$, para $i = 1, \ldots, n$. Observe que entre os M neurônios *and* da segunda camada, somente $2^{Pa} \leq M$, $Pa \leq n$, são ativos. Os neurônios correspondentes são encontrados a partir da combinação dos vetores K^1 e K^2. Isto é melhor entendido através de um exemplo. Assumindo uma rede constituida por três entradas pertencentes ao intervalo $[-2, 2]$ e uma partição com três conjuntos nebulosos para cada coordenada. Dado um padrão de entrada $\mathbf{x} = (0.5, -2, 1)$, para $x_1 = 0.5$ os conjuntos nebulosos ativos são A_1^2 e A_1^3 e para $x_2 = -2$ o conjunto nebuloso ativo é A_2^1 e para $x_3 = 1$, A_3^2 e A_3^3, formando, assim, os vetores $K^1 = (2, 1, 2)$ e $K^2 = (3, 1, 3)$. Combinando K^1 e K^2, encontra-se os quatro vetores K: $(2, 1, 2), (2, 1, 3), (3, 1, 2)$ e $(3, 1, 3)$. A seguir, usando a equação (1), são encontrados os índices j dos quatro neurônios *and* ativos: 11, 12, 20 e 21. Observe que, do total de 27 neurônios *and*, somente $Na = 2^{Pa} = 2^2 = 4$ são ativos. Também, como somente Na neurônios *and* são ativos para cada padrão de entrada apresentado, o processamento da rede é independente do número de conjuntos nebulosos das partições do espaço de entrada.

3.4 Fuzzificação

Este passo é direto, mas vale ressaltar que somente os graus de pertinência dos conjuntos nebulosos ativos (cujos índices estão em K^1) são calculados. No caso em que $k_i^1 \neq k_i^2$, $\mu_{k_i^2}(x^i)$ também é calculada. Como as funções de pertinência são complementares, tem-se que $\mu_{k_i^2}(x_i) = 1 - \mu_{k_i^1}(x_i)$. Assim, somente $2n$ graus de pertinência precisam ser determinados.

3.5 Atualização dos Pesos

A atualização dos pesos é baseada no método do gradiente e no aprendizado por reforço associativo (Barto e Jordan, 1987). O primeiro passo é calcular a saída da rede para um dado padrão de entrada $\mathbf{x}$. Isto corresponde a fuzzificação do padrão de entrada, e o cálculo sucessivo das saídas dos neurônios restantes das camadas (usando as equações (2) e (3)). O processo de aprendizagem supervisionado

tem como objetivo minimizar o valor esperado do erro entre a saída da rede atual e a saída desejada para cada padrão de entrada, ou seja, minimizar

$$\epsilon = \frac{1}{2}(y_d - y)^2 \tag{16}$$

onde y é o valor da unidade de saída e y_d é o valor desejado para o correspondente padrão de entrada $\mathbf{x}$.

Assim, a atualização dos pesos da camada de saída é feita usando o método do gradiente descendente (Rumelhart et al., 1986). Isto é, se v_j é um peso conectado à unidade de saída, então

$$\Delta v_j = \eta(y_d - y)\psi'(u)y_j \tag{17}$$

onde $\psi'(u) = \psi(u)(1-\psi(u))$ é a derivada da função de ativação avaliada em u, $u = \mathop{\mathbf{S}}_{j=1}^{M}(v_j\ \mathbf{t}\ y_j)$, e η é a taxa de aprendizagem.

Os pesos das unidades intermediárias (neurônios *and*) são atualizados usando um sinal de reforço para todas as unidades intermediárias. A regra de atualização dos pesos das unidades intermediárias depende dos valores do sinal de reforço.

Em (Barto e Jordan, 1987) foi sugerido uma variável real $\delta = 1 - \epsilon$ como sendo o sinal de reforço. Grandes valores de δ correspondem a melhores resultados entre a saída da rede e a saída desejada. Em (Caminhas et al., 1999) um esquema de recompensa ou punição foi usado para atualizar os pesos da rede. Neste artigo foi introduzido um novo esquema de atualização que, em um mesmo sentido, combina os procedimentos de atualização dos pesos propostos em (Barto e Jordan, 1987) e (Caminhas et al., 1999). Aqui, os pesos das unidades intermediárias são atualizados como segue:

$$\Delta w_{ji} = \delta\,\alpha_1\,[1 - w_{ji}] - (1-\delta)\,\alpha_2\,w_{ji} \tag{18}$$

$$\Delta r_j = \delta\,\alpha_3\,[1 - r_j] - (1-\delta)\,\alpha_4\,r_j \tag{19}$$

onde $0 < \alpha_1 << \alpha_2 < 1$ e $0 < \alpha_3 << \alpha_4 < 1$ são taxas de aprendizagem.

4 Resultados de Simulação

Nesta seção, resultados de simulação para o problema de modelagem de sistemas não-lineares são apresentados. Com o propósito de comparação, o exemplo adotado é o mesmo que sugerido em (Narendra e Parthasarathy, 1990). O processo não-linear a ser modelado é da forma:

$$y(t+1) = f\,[y(t),\ y(t-1),\ y(t-2),\ u(t),\ ,u(t-1)]$$

onde a função f, suposta desconhecida, tem a forma

$$f\,[x_1,\ x_2,\ x_3,\ x_4,\ x_5] = \frac{x_1x_2x_3x_5(x_3-1) + x_4}{1 + x_2^2 + x_3^2} \tag{20}$$

Para a fase de treinamento, o sinal de entrada $u(t)$ foi gerado aleatoriamente, com distribuição uniforme no intervalo $[-1, 1]$. A RNNR usada para modelar este sistema tem uma única entrada $u(t)$ e o número de partições para esta variável de entrada N_1 igual a 20 conjuntos nebulosos,resultando em apenas 20 neurônios na camada intermediária. Foram escolhidas como *t-norma* o produto algébrico e como *s-norma* a soma probabilística. As taxas de aprendizado usadas para atualizar os pesos da camada intermediária foram $\alpha_1 = \alpha_3 = 0.001$ e $\alpha_2 = \alpha_4 = 0.0001$. Para atualizar os pesos da camada de saída foi escolhido $\eta = 0.5$. O processo de aprendizado encerra após 173 iterações. A figura 5 mostra as saídas do sistema e da RNNR para a entrada:

$$u(t) = \begin{cases} sen(2\pi k/250), & \text{para } k \le 500 \\ 0.8sen(2\pi k/250) + & \\ 0.2sen(2\pi k/25), & \text{para } k > 500 \end{cases} \tag{21}$$

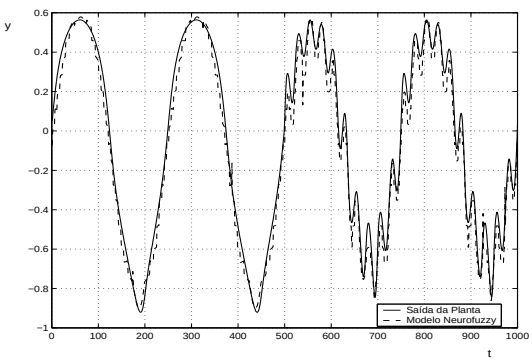

Figura 5. Saídas da planta e do modelo neural nebuloso.

Deve ser notado que em (Narendra e Parthasarathy, 1990) foram utilizadas 5 entradas ($y(t)$, $y(t-1)$, $y(t-2)$, $u(t)$, $,u(t-1)$), duas camadas intermediárias com 20 e 10 neurônios, respectivamente, tendo, portanto, 200 parâmetros ajustáveis, e 100.000 iterações para convergir. Em (Wang, 1994) o mesmo problema foi resolvido usando uma rede neural nebulosa estática com as mesmas entradas que em (Narendra e Parthasarathy, 1990) e 40 funções de pertinência Gaussianas para cada componente de entrada, correspondendo a 120 parâmetros ajustáveis, pois para cada regra existem três parâmetros ajustáveis, e 5000 iterações para convergir.

Em (Lee e Teng, 2000) o exemplo de identificação foi resolvido usando uma rede neural nebulosa estática e uma rede neural nebulosa recorrente. Para teste, o seguinte sinal de entrada $u(t)$ foi utilizado:

$$u(t) = \begin{cases} sen(\pi k/25), & 0 < k < 500 \\ 1.0, & 250 \le k < 500 \\ -1.0, & 500 \le k < 750 \\ 0.3sen(\pi k/25) + & 0.1sen(\pi k/32) \\ 0.6sen(\pi k/10), & 750 \le k < 1000 \end{cases} \tag{22}$$

A figura 6 mostra as saídas do sistema e da RNNR proposta aqui para este sinal de entrada.

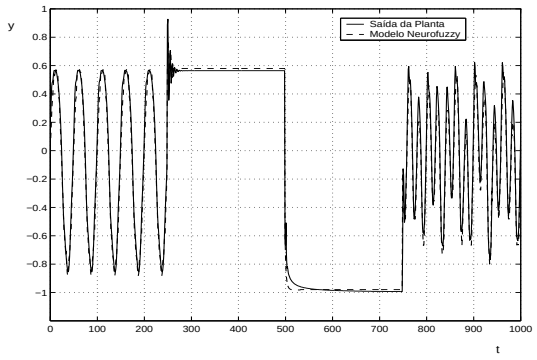

Figura 6. Saídas da planta e do modelo neural nebuloso.

No treinamento da rede neural nebulosa de (Lee e Teng, 2000), foram utilizadas 5 entradas $(y(t), y(t-1), y(t-2), u(t), , u(t-1))$, o número de regras nebulosas igual a 16, número de neurônios igual a 112 e número de parâmetros ajustados igual a 176. Na rede neural nebulosa recorrente de (Lee e Teng, 2000) foram utilizados duas entradas $(y(t), u(t))$ para modelar o sistema. O número de regras foi 16, número de neurônios igual a 51 e o número de parâmetros a ser ajustados foi igual a 112. O número de iterações foi 90.000, tanto para a rede neural nebulosa estática como dinâmica.

Como pode ser observado, a RNNR introduzida neste artigo é mais simples e eficiente que as outras mencionadas, apresentando um menor número de regras (20), conseqüentemente menor número de parâmetros (60) a serem ajustados. O número de iterações para a convergência do processo foi significativamente menor. A ordem de grandeza dos erros no conjunto de teste foi de 10^{-4} para os sinais de entrada (21) e (22).

5 Conclusões

Neste artigo foi proposto um modelo de rede neural nebulosa recorrente cujos neurônios são unidades de processamento que realizam operações consistentes com a teoria de conjuntos nebulosos. Os neurônios são unidades lógicas com operações sinápticas e somáticas definidas por *t-normas* e *s-normas*. A rede tem uma estrutura multi-camadas recorrente, e uma relação dual com o sistema baseado em regras, o qual processa informação seguindo os princípios de raciocínio aproximado. O procedimento de aprendizagem introduzido aqui é uma combinação do método do gradiente e aprendizado por reforço associativo. Os pesos da camada de saída são ajustados via um método do gradiente e os pesos da camada intermediária via uma combinação do esquema de punição e recompensa. Devido a sua natureza, o método de aprendizagem é rápido, sendo a complexidade de operações da ordem de $O(2^n)$, o qual concentra o tempo de processamento. Os resultados de simulação tem sido promissores, mostrando que a rede neural nebulosa recorrente tem alto desempenho em termos de capacidade de aproximação, memória e tempo de processamento.

Agradecimentos

Esta pesquisa teve o suporte da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) processo $\sharp 00/06198-7$, e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) processo $\sharp 300729/86-3$.

Referências Bibliográficas

Ballini, R., Soares, S. e Gomide, F. (2001). A recorrent neurofuzzy network structure and learning procedure, *10th IEEE International Conference on Fuzzy Systems- FUZZ-IEEE 2001.*

Barto, A. G. e Jordan, M. I. (1987). Gradient following without back-propagation in layered networks, *Proc. of the IEEE First International Conference on Neural Networks*, Vol. II, San Diego, pp. 629–636.

Caminhas, W., Tavares, H., Gomide, F. e Pedrycz, W. (1999). Fuzzy set based neural networks: Structure, Learning and Application, *Journal of Advanced Computational Intelligence* **3**(3): 151–157.

Ku, C. C. e Lee, K. L. (1995). Diagonal recurrent neural networks for dynamic systems control, *IEEE Trans. on Neural Networks* **6**: 144–156.

Lee, C. H. e Teng, C. C. (2000). Identification and control of dynamic systems using recurrent fuzzy neural networks, *IEEE Trans. on Fuzzy Systems* **8**(4): 3493–366.

Narendra, K. S. e Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks, *IEEE Trans. on Neural Networks* **1**(1): 4–27.

Pedrycz, W. e Gomide, F. (1998). *An Introduction to Fuzzy Sets: Analysis and Design*, MIT Press, Cambridge, MA.

Pedrycz, W. e Rocha, A. (1998). Fuzzy-set based models of neuron and knowledge-based networks, *IEEE Trans. on Fuzzy Systems* **4**(1): 254–266.

Rumelhart, D., Hinton, G. E. e Williams, R. J. (1986). Learning representations by back-propagating errors, *Nature (London)* **323**: 533–536.

Wang, L. X. (1994). *Adaptive Fuzzy Systems and Control*, Prentice-Hall.