Learning and Nonlinear Models – Revista da Sociedade Brasileira de Redes Neurais, Vol. 3, No. 1, pp. 08-18, 2005

ÍNDICE DE IGUALDADE E APRENDIZADO EM REDES NEUROFUZZY RECORRENTES EM PREVISÃO DE SÉRIES MACROECONÔMICAS

Rosangela Ballini[1], Fernando Gomide[2]
[1] Universidade Estadual de Campinas
Instituto de Economia
13083-857 Campinas – SP – Brasil
E-mail: ballini@eco.unicamp.br

[2] Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
13083-970 Campinas – SP – Brasil
E-mail: gomide@dca.fee.unicamp.br

Abstract – A novel learning algorithm for recurrent fuzzy neural network is introduced in this paper. The core of the learning algorithm uses equality index as the performance measure to be optimized. Equality index is especially important because its properties reflect the fuzzy set-based structure of the neural network and nature of learning. Equality indexes are strongly tied with the properties of the fuzzy set theory and logic-based techniques. The neural network recurrent topology is built with fuzzy neuron units and performs neural processing consistent with fuzzy system methodology. Therefore neural processing and learning are fully embodied within fuzzy set theory. The performance recurrent fuzzy neural network is verified via examples of learning sequences. Computational experiments show that the recurrent fuzzy neural models developed are simpler and that learning is faster than both, static neural and time series models.

Palavras Chaves - Redes neurais, redes neurofuzzy recorrentes, previsão de séries temporais.

1. Introdução

Redes neurais recorrentes são estruturas de processamento capazes de apresentar uma grande variedade de comportamentos dinâmicos. A presença de realimentação de informação permite a criação de conexões internas e dispositivos de memória capazes de processar e armazenar informações temporais e sinais seqüenciais.

Em contrapartida à possibilidade de representação de comportamento dinâmico, o processo de adaptação e a análise da capacidade de processamento de informação presente em redes neurais recorrentes são geralmente mais complexos que no caso não recorrente [1]. Como conseqüência, a capacidade de aprendizado desses modelos tem que estar associada com a existência de algoritmos de treinamento eficientes, baseados em informações de segunda ordem e avançados resultados da teoria de sistemas dinâmicos não-lineares [2].

Como é amplamente conhecido, a idéia de incorporar conceitos da teoria de conjuntos fuzzy em redes neurais tem se tornado um importante tópico de pesquisa [3]. Combinações destas duas abordagens, sistemas neurofuzzy, têm tido sucesso em muitas aplicações. A abordagem neurofuzzy une a teoria de conjuntos fuzzy e redes neurais em um sistema integrado para combinar os benefícios de ambas. Entretanto, uma limitação de muitas redes neurofuzzy é sua restrita aplicação em problemas de mapeamentos dinâmicos devido a sua estrutura não recorrente ou à falta de um procedimento de aprendizado para as conexões de realimentação. Sistemas neurofuzzy recorrentes têm sido propostos para identificação de sistemas dinâmicos em [4] , [5], [6].

Neste artigo, um algoritmo de aprendizado para redes neurofuzzy recorrentes é proposto. O algoritmo usa o índice de igualdade como medida de desempenho a ser otimizado. O índice de igualdade, proposto em [7], [8], preserva as propriedades básicas da teoria de conjuntos fuzzy. Como a topologia da rede neural recorrente é constituída por neurônios fuzzy, o processamento neural é consistente a teoria de conjuntos fuzzy. Portanto, o processamento neural e o aprendizado são tratados dentro do contexto da teoria de conjuntos fuzzy.

O processo de aprendizado da rede neurofuzzy proposta envolve três fases principais. A primeira fase usa uma modificação no aprendizado *vector quantization* sugerido em [9] para granular o universo de entrada. Na próxima fase, as conexões da rede são geradas e os pesos são inicializados aleatoriamente. A terceira fase usa dois principais paradigmas para atualizar os pesos da rede: método do gradiente e aprendizado por reforço associativo. Mais precisamente, os pesos da camada de saída são ajustados via um método do gradiente para maximizar o índice de igualdade e um esquema de recompensa e punição atualiza os pesos da camada intermediária.

A rede neural usa unidades neurofuzzy modeladas através de operações *and* e *or* processadas via *t-normas* e *s-normas*, respectivamente. O modelo neurofuzzy é composto por um elemento não-linear colocado em série com os neurônios fuzzy. Isto significa que cada neurônio realiza mapeamento entre o hipercubo e o intervalo unitário. A incorporação desta não linearidade modifica as características numéricas dos neurônios, mas seu comportamento lógico é mantido. Além disso, as não linearidades adicionam interpretabilidade pois podem ser tratadas como modelos de modificadores lingüísticos [8], [9], [10].
O modelo implicitamente codifica um conjunto de regras se-então em uma estrutura recorrente multicamada, realizando um processo de inferência fuzzy. A topologia sugere uma clara relação entre a estrutura da rede e um sistema baseado em regras fuzzy.

A rede neurofuzzy recorrente é particularmente adequada para modelar sistemas dinâmicos não lineares e para aprender seqüências. As relações temporais são codificadas na segunda camada da rede, fornecendo os elementos de memória e expandindo sua capacidade básica de incluir representações temporais. Como o neurônio recorrente tem uma realimentação interna, este captura respostas e contribuições para simplificar os modelos neurofuzzy.
O desempenho da rede neurofuzzy recorrente é verificado via um exemplo de previsão um passo à frente e vários passos à frente de uma série macroeconômica. Comparações com o desempenho de uma rede neural estática e com modelos de séries temporais também são apresentadas.

2. Estrutura da Rede Neural Nebulosa

As unidades de processamento consideradas aqui são neurônios denominados Neurônio-T e Neurônio-S, Figura 1, produzindo mapeamentos $[0,1]^n \rightarrow [0,1]$, [8], [11]. A implementação das operações envolve normas triangulares, ou seja, os operadores *and* e *or* são implementados através de *t-normas* e *s-normas*, resultando uma estrutura neural na qual $w_i \in [0,1]$ é o peso associado à entrada $x_i \in [0,1]$ e $\psi_{and}, \psi_{or} : [0,1] \rightarrow [0,1]$ são mapeamentos não lineares.

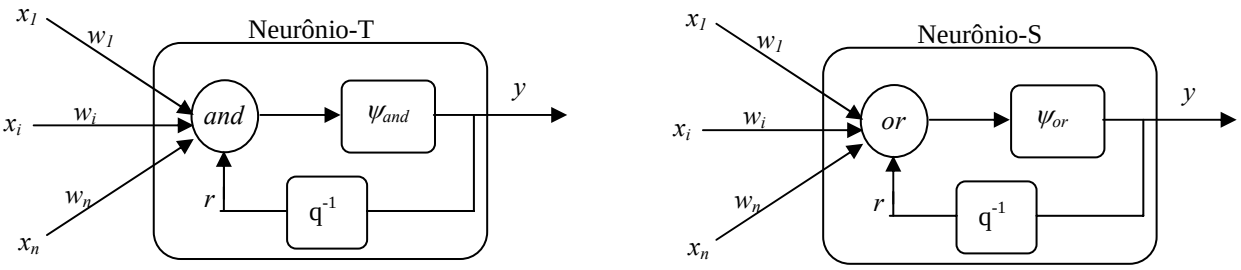

Figura 1: Neurônios fuzzy recorrentes.

Como mostra a Figura 2, a rede neural proposta neste trabalho tem uma estrutura multicamada com realimentação global nos neurônios da segunda camada. Em outras palavras, a topologia da rede permite que todos os neurônios da segunda camada estejam conectados entre si.

Mais precisamente, a rede contém três camadas. A camada de entrada consiste de neurônios cujas funções de ativação são funções de conjuntos fuzzy que formam a partição do espaço de entrada. Para cada componente $x_i(t)$ de um vetor de entrada *n*-dimensional $\mathrm{x}(t)$ existem N_i conjuntos fuzzy $A_i^{k_i}, k_i = 1,...,N_i$ cujas funções de pertinência são funções de ativação dos correspondentes neurônios da camada de entrada. A variável *t* denota o tempo discretizado, isto é, $t = 1,2,...$, 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, $a_{ji} = \mu_{A_i^{k_i}}(x_i), i = 1,...,n$ e $j=1, ..., M$, onde *M* representa o número de neurônios da segunda camada.

Note na Figura 2 que, a arquitetura inclui realimentação global na camada intermediária, cujas unidades são representadas por neurônios-T. Desta forma, os operadores *and* tem entradas a_{ji} ponderadas por w_{ji} e conexões de realimentação ponderadas por $r_{jl}, l = 1,...,M$. A função de ativação ψ_{and} é , em geral, um mapeamento não-linear cujo significado pode ser associado com um modificador lingüístico.

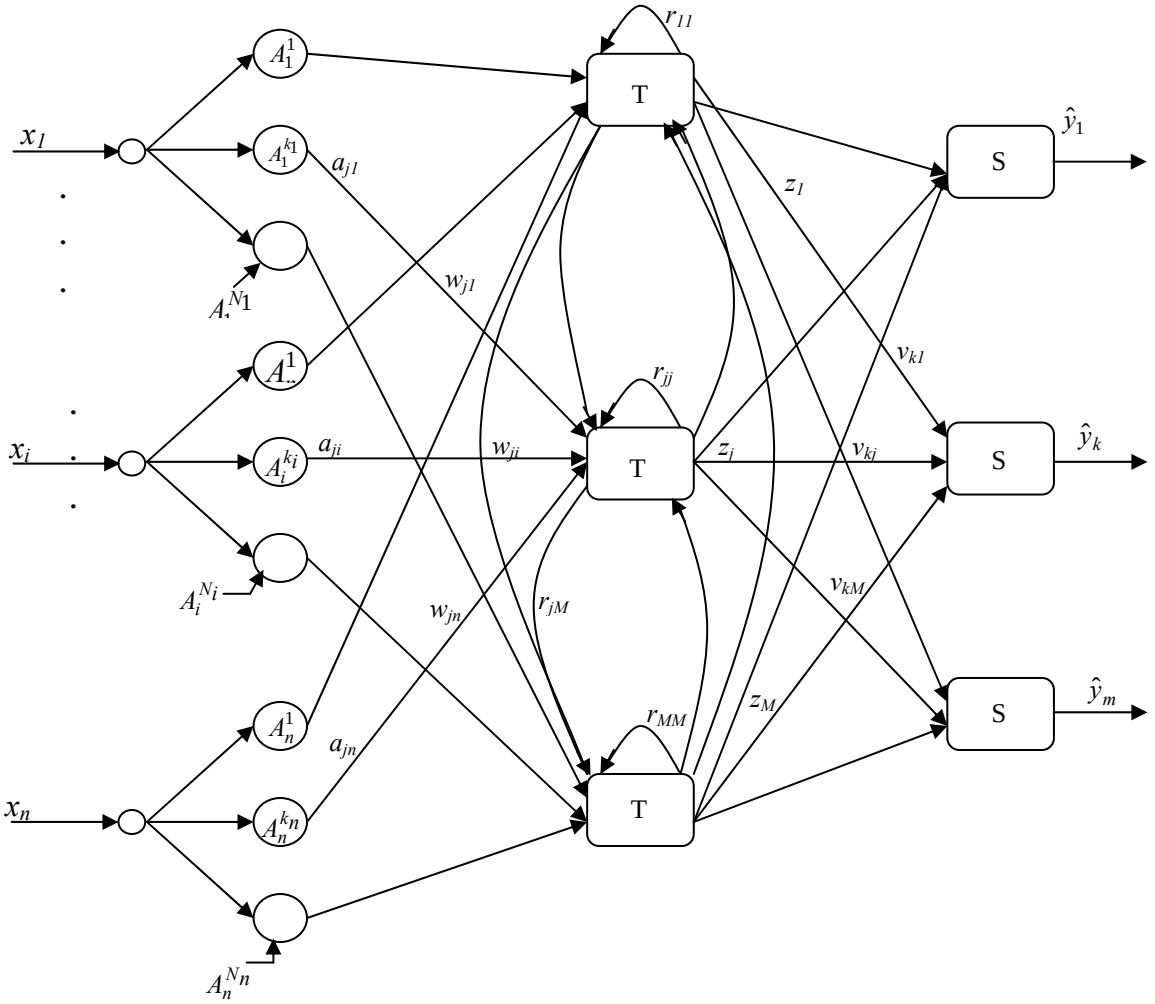

Figura 2: Estrutura da rede neurofuzzy recorrente.

Assumindo que $a_{jn+1} = q^{-1} z_j$, sendo q^{-1} o operador atraso, e $w_{jn+1} = r_{jl}$, $l = 1,...,M$, a estrutura da rede sugere um conjunto de regras se-então $R = \{R_j, j = 1,...,M\}$ da seguinte forma:

R_j: Se(x_1 é $A_1^{k_1}$ com relevância w_{j1})...E(x_i é $A_i^{k_i}$ com relevância w_{ji})...E(x_n é $A_n^{k_n}$ com relevância w_{jn}) Então z é z_j sendo $z_j = \mathop{\mathbf{T}}_{i=1}^{n+M} (w_{ji} \,\mathbf{s}\, a_{ji})$, **T** e **s** normas triangulares. Portanto, existe uma dualidade entre a primeira parte da estrutura da rede, composta pela primeira e segunda camadas, e um conjunto de regras fuzzy. O esquema de processamento das entradas pela rede segue os princípios da teoria de conjuntos nebulosos e raciocínio aproximado. Portanto, o processamento realizado pela primeira parte da estrutura da rede é equivalente aquele realizado por um sistema de inferência fuzzy [8].
A terceira camada contém *m* neurônios-S, não recorrentes. A saída deste neurônio é denotada por $\hat{y}_k$, as entradas por z_j e os pesos por v_{kj}. A função de ativação ψ_{or} dos neurônios são mapeamentos não-lineares. Observe que estes neurônios realizam operações de agregação.

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

1. N_i é o número de conjuntos fuzzy que constitui a partição do universo da *i*-ésima entrada;
2. *j* é o índice do neurônio-T. Da estrutura da rede, *j* é determinado a partir dos índices de entrada k_i como segue:
3.

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

onde, $K = (k_1, \ldots, k_i, \ldots, k_n)$;

4. $x_1, x_2, ..., x_n$ são *n* entradas da rede para um dado padrão **x**;
5. $a_{ji} = \mu_{A_i^{k_i}}(x_i)$ é o grau de pertinência de x_i no conjunto fuzzy A_i^{ki}, sendo a_{ji} a entrada para o neurônio *j* da camada intermediária;
6. w_{ji} é o peso entre a *i*-ésima entrada e o *j*-ésimo neurônio-T;
7. z_j é a saída do *j*-ésimo neurônio-T, sendo determinado por:

$$z_j = \psi_{and}\left(\mathop{\mathbf{T}}_{i=1}^{n+M} (w_{ji} \,\mathbf{s}\, a_{ji}) \right) \tag{2}$$

onde ψ_{and}: [0, 1] → [0, 1].

8. v_{kj} é o peso para a *j*-ésima entrada do *k*-ésimo neurônio-T; r_{jj} é o peso da conexão recorrente do *j*-ésimo neurônio da camada intermediária e r_{jl} é a conexão entre os *j*-ésimo e *l*-ésimo neurônios da camada intermediária.
9. $\hat{y}_k$ é à saída do neurônio não linear dada por:

$$\hat{y}_k = \psi_{or}(u_k) = \psi_{or}\left(\mathop{\mathrm{S}}_{j=1}^{M} (v_{kj} \,\mathrm{t}\, z_j) \right) \tag{3}$$

onde ψ_{or}: [0, 1] → [0, 1] é uma função não-linear. Neste trabalho supomos que ψ_{or} é a função logística $1/(1+exp(-u_k))$, ψ_{and} a função identidade, e que as t-normas e s-normas são o produto algébrico e a soma probabilística, respectivamente:

$$x \,\mathbf{t}\, y = xy \quad \text{e} \quad x \,\mathbf{s}\, y = x + y - xy \tag{4}$$

3. Processo de Aprendizagem

O esquema de aprendizado para a rede neurofuzzy recorrente envolve três fases. A primeira fase usa uma modificação do procedimento *vector quantization* para granular o universo de entrada. A próxima fase simplesmente gera as conexões da rede e inicializa os pesos aleatorimente. A terceira fase utiliza dois conceitos diferentes para o ajuste dos pesos: aprendizagem por reforço associativo e método do gradiente.

O esquema de aprendizado é primeiro apresentado, de uma forma geral, no Quadro 1. Os passos essenciais para implementação são detalhados nas seções seguintes. O algoritmo é uma forma de aprendizado supervisionado.

Quadro 1: Algoritmo de Aprendizado da Rede NeuroFuzzy Recorrente

1. Gerar as funções de pertinência;
2. Inicializar os pesos w_{ji}, v_{kj} e r_{ji};
3. Até que a condição do critério de parada não seja satisfeita fazer:
   - 3.1 Apresentar um padrão **x** à rede;
   - 3.2 Efetuar a fuzzificação;
   - 3.3 Determinar os neurônios-T ativos;
   - 3.4 Atualize os pesos w_{ji}, v_{kj} e r_{jl};

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, assumimos funções triangulares, definidas no universo $[x_{i_{\min}}, x_{i_{\max}}], i = 1, ..., n$.

A forma mais simples de gerar funções de pertinência é construir partições uniformes e complementares. Intuitivamente, se na Figura 3 os triângulos intermediários forem todos isósceles, iguais, com altura unitária, e os triângulos nas extremidades forem similares, mas retângulos, então a partição seria uniforme. Observe que, no caso ilustrado na Figura 3, para cada ponto do universo somente duas funções de pertinência tem valores não nulos e que a soma dos correspondentes graus de perinência é sempre igual à unidade. Neste caso, a partição é complementar. Contudo esta pode não ser a mais adequada se um número de padrões está concentrado em certas regiões do universo e dispersas em outras. Nestes casos, partições não uniformes são mais apropriadas, como é o caso da Figura 3.

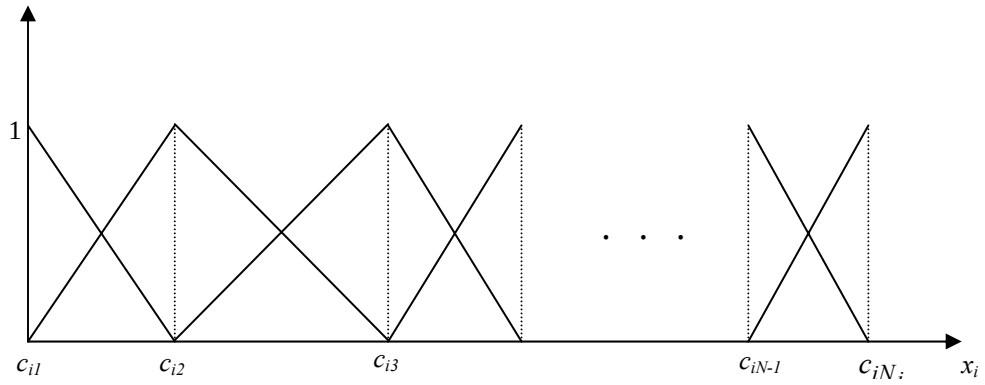

Figura 3: Partições não uniformes.

Com este propósito, técnicas de agrupamento para encontrar os centros das funções c_{ir}, $r=1, \ldots, N_i$, são utilizadas. Em [9], uma variação de uma rede neural auto-organizada (Figura 4) com algoritmo de treinamento tipo LVQ (*Learning Vector Quantization*) é sugerida.

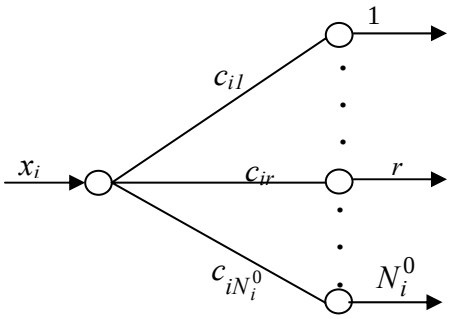

Figura 4: Rede neural auto-organizada para determinar os centros das funções de pertinência.

Na estrutura da Figura 4, os pesos da rede são centros de grupos c_{ir}. O treinamento da rede neural auto-organizada da Figura 4 é não supervisionado e competitivo. Somente o peso da conexão do neurônio vencedor é ajustado. Ao final do treinamento, aqueles neurônios que tiveram um baixo índice de desempenho, isto é, venceram poucas vezes, são eliminados da rede. Este procedimento permite determinar o número de funções de pertinência adequado, N_i, e seus respectivos centros. O algoritmo é o seguinte.

Algoritmo Para Geração das Funções de Pertinência

1. Inicializações:
   1.1 Pesos c_{ir}:

$$c_{i1} = x_{imin} \tag{5}$$

$$c_{ir} = c_{i(r-1)} + \Delta_i, \text{ para } r=2, 3, \ldots, N_i^0 . \tag{6}$$

sendo, $\Delta_i = \frac{x_{i\max} - x_{i\min}}{N_i^0 - 1}$.

   1.2 Índice de Desempenho: $id_i(r)=0$, para $r=2, 3, \ldots, N_i^0$, ou seja, as funções de pertinência iniciais são partições uniformes. Este procedimento de inicialização geralmente proporciona uma convergência mais rápida que a inicialização aleatória;
2. Até que $\left|c_{ir}(t+1) - c_{ir}(t)\right| \le \varepsilon, \forall j$ fazer:
   2.1 Apresentar um padrão x_i à rede e atualizar o peso da conexão do neurônio vencedor da seguinte forma:

$$c_{iL}(t+1) = c_{iL}(t) + \alpha(t)\left[x_i - c_{iL}(t)\right] \tag{7}$$

sendo L o índice do neurônio vencedor, que é aquele cujo peso da conexão possui o valor mais próximo do padrão x_i, ou seja,

$$L = arg\left\{ \min_{r} |x_i - c_{ir}| \right\} \tag{8}$$

2.2 Decrescer o tamanho do passo $\alpha(t)$;
2.3 Atualizar o índice de desempenho do neurônio vencedor, da seguinte forma:

$$id_i(L) = id_i(L) + 1 \tag{9}$$

3. Eliminar todos os neurônios cujo valor de $id_i \leq \delta$, sendo δ um inteiro positivo. Seja Nne_i o número de neurônios eliminados para a coordenada i. Assim, o número de conjuntos *fuzzy* para a coordenada i é:

$$N_i = N_i^0 - Nne_i \tag{10}$$

4. Fim.

Uma vez encontrados o número de conjuntos fuzzy N_i e os centros c_{ir}, $i = 1, \ldots, n$ e $r = 1, \ldots, N_i$, o cálculo das funções de pertinência é bastante simples, envolvendo duas operações de soma e uma de multiplicação:

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

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

Após a geração das funções de pertinências, o próximo passo é a inicialização dos pesos da rede. Os neurônios da primeira e da segunda camada estão inteiramente conectados. Os pesos w_{ji}, v_{kj} e r_{jl}, para $i = 1, \ldots, n$, $j = 1, \ldots, M$ e $l = 1, \ldots, M$, que representam os pesos da camada de entrada, camada de saída e das conexões de realimentação, respectivamente, são inicializados aleatoriamente com valores entre [0, 1].

3.2 Determinação dos neurônios-T ativos

Para cada padrão de entrada **x**, 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-T ativos, sendo facilmente identificados como segue.

Dado um 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 função 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

$$k_i^2 = \begin{cases} k_i^1 + 1, & \text{se } \mu_{A_i^{k_i^1}}(x_i) \neq 1 \\ k_i^1, & \text{caso contrário} \end{cases} \tag{12}$$

O número de neurônios-T ativos é $Na = 2^{Pa} \leq 2^n$, sendo Pa o número de elementos tal que $k_i^1 \neq k_i^2$, para $i = 1, \ldots, n$. Observe que entre os M neurônios-Tda 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. Também, pode-se notar que, como somente Na neurônios-T são ativos para cada padrão de entrada apresentado, o processamento da rede é independente do número de conjuntos fuzzy das partições do espaço de entrada. Isto é mais bem entendido via um exemplo.

Seja uma rede com 3 entradas definidas no intervalo [-2, 2] e uma partição com 3 conjuntos fuzzy para cada coordenada do vetor de entrada. Dado um padrão de entrada **x**=(0.5, -2, 1), para a coordenada $x_1 = 0.5$ os conjuntos fuzzy ativos são A_1^2 e A_1^3, para $x_2 = -2$, o conjunto ativo é A_2^1 e para $x_3 = 1$, A_3^2 e A_3^3, definido assim os vetores $K^1 = (2,1,2)$ e $K^2 = (3,1,3)$. Combinado os vetores K^1 e K^2, chega-se aos seguintes vetores: (2, 1, 2), (2, 1, 3), (3, 1, 2) e (3, 1, 3), que, a partir da expressão (1), permitem identificar os 4 neurônios-T indexados por "j", iguais a: 11, 12, 20, 21, como neurônios ativos. Observe que, do

conjunto de 27 neurônios-T, somente $Na = 2^{Pa} = 2^2 = 4$ são ativos. A vantagem do emprego desse passo do algoritmo é que somente os neurônios ativos serão considerados para efeito de cálculo nos passos seguintes. Com isto o tempo de processamento do algoritmo fica independente do número de partições do espaço de entrada, ou seja, independente do número de neurônios-T.

3.3 Fuzzificação

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

3.4 Atualização dos pesos

A atualização dos pesos é baseada no método do gradiente e no aprendizado por reforço associativo [12]. O método do gradiente é usado para ajustar os pesos da rede neural, isto é, este método é usado para atualizar os pesos da camada de saída. O aprendizado por reforço associativo é utilizado para ajustar os pesos da camada intermediária, ou seja, do sistema de inferência fuzzy.

O primeiro passo é calcular a saída da rede para um dado padrão de entrada **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 algoritmo de aprendizado usa o índice de igualdade como a medida de desempenho a ser otimizada. O critério de índice de igualdade é definido como em [7]:

$$Q_j = \sum_{k=1}^{N} (y_{kj} \equiv \hat{y}_{kj}) \tag{13}$$

onde, $\hat{y}_{kj}$ é o valor da unidade de saída j e y_{kj} é o valor desejado j para o correspondente padrão de entrada **x**. Formalmente, a utilização desse critério como medida de desempenho a ser otimizada, é devido este critério preservar as propriedades básicas da teoria de conjuntos *fuzzy*. eComo a topologia da rede neural proposta é constituída por neurônios *fuzzy*, o processamento neural deve ser consistente com a teoria de conjuntos *fuzzy*.

O valor do índice de igualdade é calculado como segue:

$$y_{kj} \equiv \hat{y}_{kj} = \begin{cases} 1 + y_{kj} - \hat{y}_{kj}, & \text{se } y_{kj} < \hat{y}_{kj} \\ 1 + \hat{y}_{kj} - y_{kj}, & \text{se } y_{kj} > \hat{y}_{kj} \\ 1, & \text{se } y_{kj} = \hat{y}_{kj} \end{cases} \tag{14}$$

Intuitivamente, partir de (14) vemos que $y_{kj} \equiv \hat{y}_{kj}$ é a unidade se e somente se $y_{kj} = \hat{y}_{kj}$ e que, ao contrário dos índices clássicos como erro quadrático tradicional, sabemos o seu valor ótimo quando o conjunto de treinamento é finito.

A soma em (13) é dividida em três grupos, dependendo da relação entre y_{kj} e $\hat{y}_{kj}$:

$$Q_j = \sum_{k:\hat{y}_{kj} > y_{kj}} (y_{kj} \equiv \hat{y}_{kj}) + \sum_{k:\hat{y}_{kj} = y_{kj}} (y_{kj} \equiv \hat{y}_{kj}) + \sum_{k:\hat{y}_{kj} < y_{kj}} (\hat{y}_{kj} \equiv y_{kj}) \tag{15}$$

Assim, a atualização dos pesos da camada de saída é feita usando o método do gradiente descendente [13]. Isto é, se v_{kj} é um peso conectado à unidade de saída do neurônio j à entrada do neurônio k, então

$$\Delta v_{kj} = \frac{\eta(t)}{N} \left[\frac{\partial Q_j}{\partial v_{kj}(t+1)} + \alpha \frac{\partial Q_j}{\partial v_{kj}(t)} \right] \tag{17}$$

onde $\eta(t)$ é a taxa de aprendizado, dada por:

$$\eta(t) = \frac{1}{\sqrt{t} + 10} \tag{18}$$

sendo o índice *t* a iteração, *t*=1, 2, . . . e α é o termo momentum. A derivada $\frac{\partial Q_j}{\partial v_{kj}}$ é escrita como

$$\frac{\partial Q_j}{\partial v_{kj}} = -\sum_{k:\hat{y}_{kj} > y_{kj}} \frac{\partial \psi_{or}(.)}{\partial v_{kj}} + \sum_{k:\hat{y}_{kj} < y_{kj}} \frac{\partial \psi_{or}(.)}{\partial v_{kj}} \tag{19}$$

$\psi_{or}(.)$ é dado por (3). Todas as derivadas podem ser computadas quando a forma da função ψ_{or} é especificada. Neste caso, a função ψ_{or} é a função logística.

Note que os incrementos são positivos quando $\hat{y}_{kj}$ é menor que o valor desejado y_{kj} e negativo quando a situação inversa ocorre. Assim, todos os incrementos são direcionados pelo sinal da diferença entre $\hat{y}_{kj}$ e y_{kj}.

Os pesos das unidades intermediárias (neurônios-T) 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 δ.

Em [9] um esquema de punição e recompensa foi usado para atualizar os pesos e a eficiência do modelo foi demonstrada através do problema de classificação de padrões.Neste caso, se a rede classifica corretamente um padrão de entrada, os pesos mais relevantes são incrementados. Caso contrário os pesos são decrementados.

No caso de classificação de padrões atribui-se "classe correta" ou "sucesso" para $\delta = 1$ e "classe incorreta" ou "fracasso" se $\delta = 0$, e para cada unidade intermediária *i* os pesos são atualizados de acordo com a seguinte equação:

$$\Delta w_{ji} = \begin{cases} \alpha_1 [1 - w_{ji}], & \text{se } \delta = 1 \\ -\alpha_2 w_{ji}, & \text{se } \delta = 0 \end{cases} \tag{20}$$

Em [12] uma variação do algoritmo associativo recompensa-penalidade (*associative reward-penalty algorithm* – A_{rp}) [14] utiliza um valor contínuo para o sinal de reforço, $\delta = 1 - \varepsilon$, onde ε é a medida de desempenho da rede. Observe que $0 \leq \delta \leq 1$, isto é, grandes valores de δ correspondem a melhores resultados entre a saída da rede e a saída desejada.

Em [6] foi introduzido um novo esquema de atualização dos pesos, em um mesmo sentido, combinando os procedimentos de atualização dos pesos propostos em [9] e [12]. Aqui, um esquema similar é usado para atualizar os pesos das unidades intermediárias como segue:

$$\Delta w_{ji} = \delta\, \alpha_1\, (1 - w_{ji}) - (1 - \delta)\, \alpha_2\, w_{ji} \tag{21}$$

onde, $0 < \alpha_1 << \alpha_2 < 1$ e são taxas de aprendizagem, $j = 1, \ldots, M$ e $i = 1, \ldots, n+M$ e $w_{jn+l} = r_{jl}$, $l=1, \ldots, M$. O sinal de reforço é $\delta = (y_{kj} \equiv \hat{y}_{kj})$ dado por (14).

Note que o aprendizado por reforço é um processo de pesquisa com o objetivo de maximizar o valor esperado de uma função critério conhecida como sinal de reforço [12]. O esquema de punição e recompensa como em (21) é tal que se δ é próximo de um os pesos dos neurônios ativos correspondentes são incrementados. Caso contrário, os pesos dos neurônios ativos são decrementados. Claramente, a atualização dos pesos muda entre os casos extremos controlado por (20) o qual pode ser visto como uma particularidade de (21) no caso de reforço binário.

4. Resultados de Simulação

Para mostrar o desempenho obtido com o modelo proposto de rede neurofuzzy recorrente (RNFN), para aprender relações temporais, um problema de predição de uma série macroeconômica é considerado a seguir. Nesse sentido, será analisada a série do produto interno bruto real extraída da economia brasileira. A série se inicia no primeiro trimestre de 1980 e se estende até o segundo semestre de 1997, como mostra a figura 4. Estes dados foram analisados e testados com o modelo de série temporal ARIMA e com uma rede neural multicamada com algoritmo de retropropagação, como pode ser visto em [15].

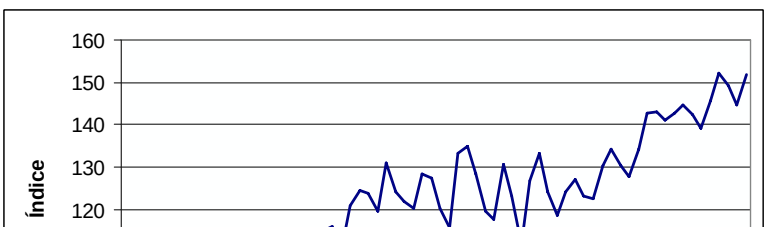

Figura 5: Produto interno bruto real.

Para ajuste do modelo somente 66 amostras da seqüência foram fornecidas, ou seja, foram utilizados os dados a partir do primeiro trimestre de 1980 até o segundo trimestre de 1996. O objetivo foi prever os próximos 4 trimestres (terceiro trimestre de 1996 ao segundo trimestre de 1997). A Figura 6 mostra a previsão 1 passo à frente e a Figura 7 mostra a previsão 4 passos à frente.

Para esta série, foram ajustadas quatro redes neurais recorrentes, ou seja, uma para cada trimestre. Cada um dos modelos RFNR usados para modelar este processo tem uma entrada $x(t)$. Foi considerada uma partição não-uniforme com N_1=16 conjuntos *fuzzy*, o qual significam 16 neurônios na camada intermediária e 288 parâmetros para ajustar. Essa estrutura foi à mesma para cada um dos trimestres.

As *t-norma* e *s-norma* escolhidas foram o produto algébrico e soma probabilística, definidas pela expressão (4). As taxas de aprendizado usadas para ajustar os pesos da camada intermediária foram α_1=0,01 e α_2 =0,001. Os valores do índice de desempenho tornam-se estáveis, assumindo o valor $Q_j = 59{,}07$ após 1000 iterações.

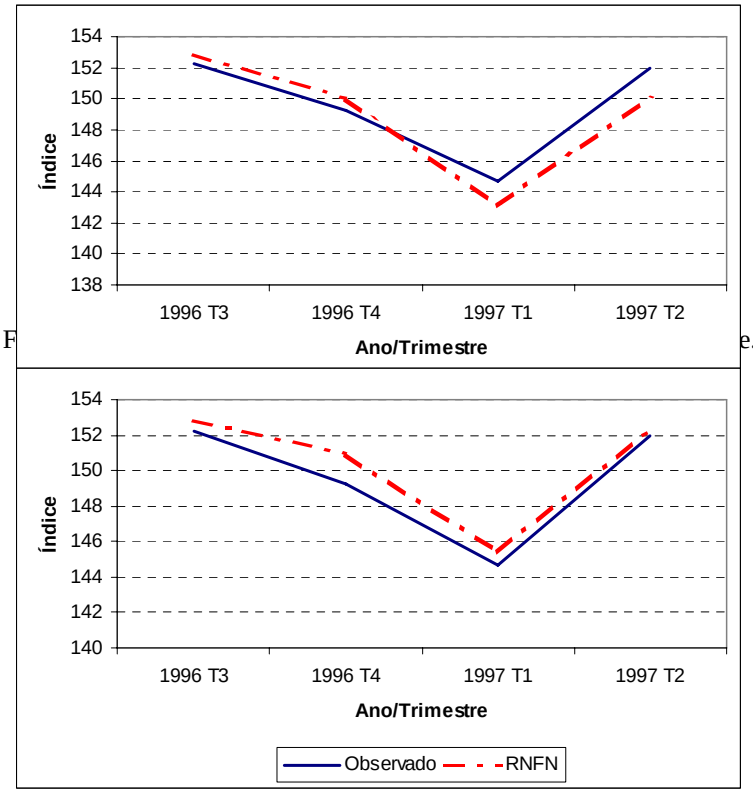

Figura 7: Predição da série produto interno bruto real, 4 passos à frente.

Como mostra a Figura 5, a série apresenta uma pequena tendência de crescimento na economia brasileira no período de 1980 a 1997. Da mesma forma há um padrão sazonal bem definido, que está associado a períodos de aquecimento da economia no terceiro trimestre e de queda da atividade econômica no primeiro trimestre de cada ano. Desta forma, o modelo de Box & Jenkins [16] estimado em [15] foi um modelo ARIMA(0, 1, 2)(0, 1, 4).

Em [15], uma rede neural foi ajustada sendo que os vetores de entrada contem 5 neurônios, ou seja, contêm o valor $x(t)$ da série e 4 neurônios binários, representando o componente sazonal (1000 para o primeiro semestre, 0100 para o segundo semestre, 0010 para o terceiro semestre e por fim 0001 para o quarto semestre). A rede possui uma camada intermediária com 4 neurônios.

A Tabela 1 resume a raiz do erro médio quadrático (*root mean square error -RMSE*), o erro médio absoluto (*mean absolute error - MAE*) e o erro médio percentual (*EMP (%)*).

Tabela 1: Desempenho dos modelos de previsão.

Modelo	1 passo à frente			4 passos à frente		
	RMSE	*EAM*	*EPM (%)*	*RMSE*	*EAM*	*EPM (%)*
ARIMA	2,87	2,20	1,12	2,52	1,79	1,23
MLP	2,23	2,01	1,41	1,82	1,30	0,89
RNFR	1,31	1,19	0,79	0,96	0,83	0,56

Observe que, tanto para previsão 1 passo à frente como para previsão 4 passos à frente, o melhor modelo é o RNFR proposto, cujos erros são significativamente menores do que todas as demais abordagens. Além disso, deve-se notar que os modelos de redes neurais apresentam melhor desempenho que o modelo ARIMA, principalmente para horizontes de previsão mais longos.

5. Conclusões

Neste artigo foi introduzido um algoritmo de aprendizado para uma estrutura de rede neurofuzzy recorrente. O algoritmo usa o índice de igualdade como medida de desempenho. O índice de igualdade é especialmente importante devido às propriedades refletirem a estrutura de conjunto fuzzy da rede neural e a natureza de aprendizado. Nas redes neurofuzzy recorrentes, os neurônios são unidades computacionais que desempenham operações consistentes com a teoria de conjuntos fuzzy. Os neurônios são unidades de processamento com operações definidas por *t-normas* e *s-normas*. A rede tem uma estrutura multicamada e é isomórfica a um sistema baseado em regras, processando informações seguindo os princípios de raciocínio aproximado. O procedimento de aprendizado introduzido é uma combinação do método do gradiente e um aprendizado por reforço associativo. Os pesos da camada de saída são ajustados via um método do gradiente e os pesos da camada intermediária são ajustados via uma heurística baseada em um esquema de recompensa e punição. O desempenho do modelo proposto foi verificado através da predição de uma série macroeconômica, mais precisamente, a série do produto interno bruto real brasileiro relativo ao período que vai do primeiro trimestre de 1980 até o segundo semestre de 1997. Os resultados foram comparados com um modelo linear de série temporal clássico, ARIMA, e um modelo não linear na forma de uma redes neural multicamada trinada com o algoritmo *backpropagation*. O desempenho da rede neurofuzzy foi melhor tanto para previsão um passo à frente como para previsão quatro passos sob o ponto de vista de todos os critérios aqui considerados, isto é, o da raiz do erro médio quadrático, o erro médio absoluto e o erro médio percentual. Além disso, os resultados também mostram que a rede neurofuzzy apresenta um melhor desempenho em termos de capacidade de aproximação funcional. Experimentos com o problema de predição de série macroeconômica sugerem que a rede neurofuzzy também é atrativa sob o ponto de vista computacional devido aos seus modestos requisitos de memória e tempo de processamento.

6. Agradecimentos

Os autores agradecem o apoio da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) Processo 2003/10019-9 e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) Processo 304299/2003-0.

7. Referências

[1] R. J. Williams e D. Zipser, "A learning algorithm for continually running fully recurrent neural networks", *Neural Computation*, vol. 1, no. 2, pp. 270-280, 1989.

[2] F. J. Von Zuben, *Modelos Paramétricos e Não-Paramétricos de Redes Neurais Artificiais e Aplicações*, Teste de Doutorado, Faculdade de Engenharia Elétrica e de Computação, Unicamp, 1996.

[3] A. Buckley e Y. Hayashi, "Fuzzy neural networks: a survey", *Fuzzy Sets and Systems*, vol. 66, pp. 41-49, 1994.

[4] A. Nürnberger, A. Radetzky e R. Kruse, "Using recurrent neuro-fuzzy techniques for the identification and simulation of dynamic systems", *Neuro-computing*, vol. 36, pp. 123-147, 2001.

[5] C. H. Lee e C. C. Teng, "Identification and control of dynamic systems using recurrent fuzzy neural networks", *IEEE Trans. On Fuzzy Systems*, vol. 8, no. 4, pp. 349-366.

[6] R. Ballini e F. Gomide, "Heuristic learning in recurrent neural fuzzy networks", *Journal of Intelligentt & Fuzzy Systems*, vol. 13, no. 2-4, 2002/2003, pp. 63-74.

[7] W. Pedrycz, "Neurocomputation in relational systems", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 289-297, 1991.

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

[9] W. Caminhas, H. Tavares, F. Gomide e W. Pedrycz, "Fuzzy set based neural networks: Structure, Learning and Application", *Journal of Advanced Computational Intelligence*, vol. 3, no. 3, pp. 151-157, 1999.

[10] R. Ballini e F. Gomide: "Learning in recurrent, hybrid neurofuzzy networks", in *Proc. 11th IEEE International Conference on Fuzzy Systems – FUZZ-IEEE'2002, IEEE World Conference – WCCI'2002*, Hawaii, USA, 2002, pp. 785 – 791.

[11] W. Pedrycz e A. Rocha, "Fuzzy-set based models of neuron and knowledge-based networks", *IEEE Trans. on Fuzzy Systems*, vol. 4, no. 1, pp. 254-266, 1998.

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

[13] D. Rumelhart, G. E. Hinton e R. J. Williams, "Learning representations by back-propagation errors", *Nature (London)*, vol. 323, pp. 533-536, 1986.

[14] A. G. Barto e P. Anandan, "Pattern recognizing stochastic learning automata", *IEEE Trans. On Systems, Man and Cybernetics*, vol. 15, pp. 360-375, 1985.

[15] G. H. A. Gontijo, *Previsão de Séries Macroeconômicas Agregadas no Ciclo de Negócios: Modelos Univariados de Séries Temporais e Redes Neurais Artificiais*, Tese de Mestrado, Faculdade de Ciências Econômicas, UFRG, 1999.

[16] G. E. P. Box, G. M. Jenkins e G. C. Reinsel, *Time Series Analysis: Forecasting and Control*. Holden Day, 3th Edição, 1994.