Proceedings of the VI Brazilian Conference on Neural Networks - VI Congresso Brasileiro de Redes Neurais
pp. 325–330, June 2–5, 2003 - UniFEI, São Paulo - SP - Brazil

Índice de Igualdade e Aprendizado em Redes Neurofuzzy Recorrentes Aplicado a Previsão de Séries Macroeconômicas

Rosangela Ballini[1], Fernando Gomide[2]

[1]Instituto de Economia
Universidade Estadual de Campinas
13083-970 Campinas - SP - Brasil
E-mail: ballini@eco.unicamp.br

[2]Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas
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 example of learning sequences. Computational experiments show that the recurrent fuzzy neural model achieves high performance in terms of approximation capability, memory requeriments, and learning speed.

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 representações 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 uma conseqüencia, a capacidade de aprendizado desses modelos tem que estar associado com a existência de algoritmos de treinamento eficientes, baseado em informações de segunda ordem e avançados resultados da teoria de sistemas dinâmicos não-lineares [2].

Em anos recentes, o conceito de incorporar lógica fuzzy e redes neurais tem tornado um importante tópico de pesquisa [3]. Combinações destas duas abordagens, sistemas neurofuzzy, têm tido sucesso em muitas aplicações. 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 proposto 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 [7], [8] como medida de desempenho a ser otimizada. O índice de igualdade é especialmente importante devido a suas propriedades refletirem a estrutura básica de conjunto fuzzy da rede neural e a natureza de aprendizado. A topologia da rede neural recorrente é constituida por neurônios fuzzy, realizando processamento neural consistente a teoria de conjuntos fuzzy. Portanto, o processamento neural e o aprendizado estão em completo acordo dentro da teoria de conjuntos fuzzy e a metodologia empregada.

O processo de aprendizado da rede 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 aleatóriamente. 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 lógicas *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 lógicos. Isto significa que cada neurônio realiza mapeamento entre as unidades do hipercubo unitário. A incorporação desta não linearidade muda 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 [9], [10].

O modelo implícitamente codifica um conjunto de regras se-então na 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, esta 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 são realizadas. Os resultados mostram que o modelo proposto apresenta alto desempenho quando comparado com as outras abordagens.

2. Estrutura da Rede Neural Nebulosa

As unidades de processamento consideradas aqui são neurônios lógicos denominados *and* e *or*, produzindo mapeamentos $[0,1]^n \rightarrow [0,1]$ [8], [11]. A implementação dos conectivos lógicos envolvem 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 a entrada $x_i \in [0,1]$ e $\psi_{and},\ \psi_{or} : [0,1] \rightarrow [0,1]$ são mapeamentos não lineares (Figura 1).

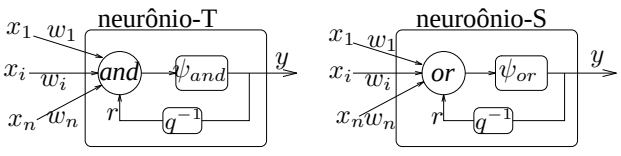

Figura 1: Neurônios lógicos recorrentes.

A rede tem uma estrutura multicamada com realimentação na saída dos neurônios da segunda camada, Figura 2.

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

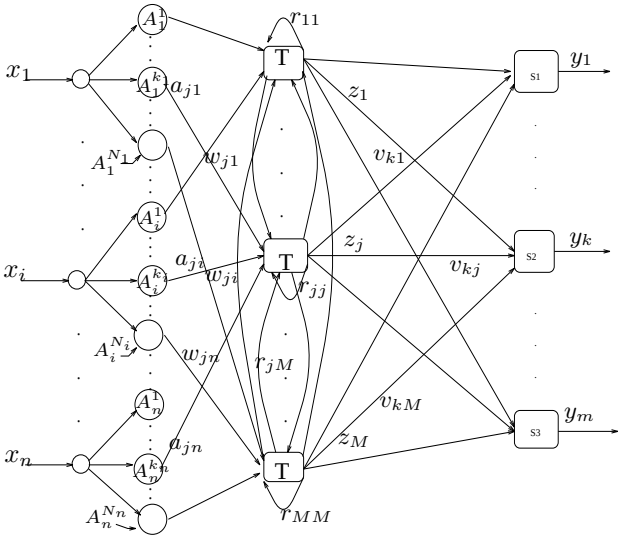

Figura 2: Estrutura da rede neurofuzzy recorrente.

espaço de entrada. Para cada componente $x_i(t)$ de um vetor de entrada n-dimensional $\mathbf{x}(t)$ existem N_i conjuntos fuzzy $A_i^{k_i},\ k_i = 1,\ldots,N_i$ cujas funções de pertinência são funções de ativação dos correspondentes neurônios da camada de entrada. A variável t denota o tempo discretizado, isto é, $t = 1,2,\ldots$, 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 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 $r_{jl}, l = 1,\ldots,M$. A função de ativação ψ_{and} é , em geral, um mapeamento não-linear cujo significado pode ser associado com um modificador lingüístico.

Assumindo que $a_{jn+1} = q^{-1}z_j$ (q^{-1} é o operador atraso) e $w_{jn+1} = r_{jl}, l = 1,\ldots,M$, 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 z_j

onde $z_j = \overset{n+M}{\underset{i=1}{\mathbf{T}}} (w_{ji}\, \mathbf{s}\, a_{ji})$. 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 fuzzy e raciocínio aproximado. A primeira parte da estrutura da rede, a qual representa o sistema de inferência fuzzy, é encerrada nesta fase.

A terceira camada contêm m neurônios-S, não recorrentes. As saídas destes neurônios são denotadas por

$\widehat{y}_k$, 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 da i-ésima entrada;
2. j é o índex do neurônio-T. Da estrutura da rede, j é determinado a partir dos índices de entrada k_i como segue:

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

onde $K = (k_1 \ \ldots \ k_i \ \ldots \ k_n)$.
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. z_j é a saída do j-ésimo neurônio-T, sendo determinado por:

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

onde $\psi_{and} : [0,1] \rightarrow [0,1]$.
7. v_{kj} é o peso para a j-ésima entrada do k-ésimo neurônio, 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.
8. $\widehat{y}_k$ é a saída do neurônio não linear dada por:

$$\widehat{y}_k = \psi_{or}(u_k) = \psi_{or} \left(\mathop{\mathbf{S}}_{j=1}^{M} (v_{kj} \ \mathbf{t} \ z_j) \right) \qquad (3)$$

onde $\psi_{or} : [0,1] \rightarrow [0,1]$ é uma função não-linear. Neste trabalho, foi considerado ψ_{or} como sendo a função logística $1/(1 + exp(-u_k))$ e ψ_{and} a função identidade.

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. Os passos essenciais para implementação serão detalhados a seguir. O algoritmo é uma forma de aprendizado supervisionado.

Algoritmo de Aprendizado da Rede NeuroFuzzy Recorrente:

1. Gere as funções de pertinência;
2. Inicialize os pesos w_{ji}, v_{kj} e r_{jl};
3. Até que a condição do critério de parada não seja satisfeita faça:
   3.1 Apresente um padrão $\mathbf{x}$ à rede;
   3.2 Efetue a fuzzificação;
   3.3 Determine 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, foram assumidas funções triangulares, definidas pelos parâmetros $[x_{i_{min}}, \ x_{i_{max}}], \ i = 1, \ldots, n$.

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).

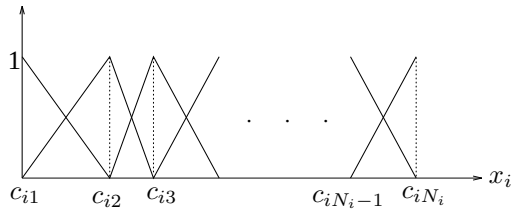

Figura 3: Partições não uniformes.

Técnicas de *clusterização* para encontrar os centros das funções $c_{ir}, \ r = 1, \ldots, N_i$ são utilizadas. Em [9] uma variação de uma rede neural auto-organizada com algoritmo de treinamento tipo LVQ (*Learning Vector Quantization*) é sugerida. Os detalhes do processo de agrupamento pode ser encontrado em [9], [10].

Uma vez determinados os centros $c_{ir}, \ i = 1, \ldots, n$ e $r = 1, \ldots, N_i$, onde N_i é o número de funções de pertinência para a entrada x_i, os graus de pertinência $\mu_{A_i^r(x_i)}$ são avaliados para cada padrão de entrada $\mathbf{x}$.

3.2. Inicialização dos pesos

Os neurônios da primeira e da segunda camada estão inteiramente conectados. Os pesos w_{ji}, v_{kj} e r_{jl} para $i = 1, \ldots, n; \ j, l = 1, \ldots, M; \ k = 1, \ldots, M$ são inicializados aleatoriamente com valores entre $[0, 1]$.

3.3. Determinação dos neurônios ativos

Para cada padrão de entrada $\mathbf{x}$, 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 *and* ativos, sendo facilmente identificados como segue.

Para cada padrão $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_n)$, 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

$$k_i^2 = \begin{cases} k_i^1 + 1, & \text{se } \mu_{A_i^{k_i^1}}(x_i) \neq 1 \\ k_i^1, & \text{caso contrário} \end{cases} \quad (4)$$

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. Também, pode-se notar que, como somente Na neurônios *and* 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.

3.4. Fuzzificação

Este passo é direto, mas vale ressaltar que somente os graus de pertinência dos conjuntos fuzzy ativos (cujos índices estão em K^1) são calculados. No caso em que $k_i^1 \neq k_i^2,\ \mu_{A_i^{k_i^2}}(x_i)$ 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.

3.5. 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 $\mathbf{x}$. 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 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] :

$$Q_j = \sum_{k=1}^{N} (y_{kj} \equiv \widehat{y}_{kj})^2 \quad (5)$$

onde $\widehat{y}_{kj}$ é o valor da unidade de saída j e y_{kj} é o valor desejado j para o correspondente padrão de entrada $\mathbf{x}$.

O valor do índice de igualdade é calculado como segue:

$$y_{kj} \equiv \widehat{y}_{kj} = \begin{cases} 1 + y_{kj} - \widehat{y}_{kj}, & \text{se } y_{kj} < \widehat{y}_{kj} \\ 1 + \widehat{y}_{kj} - y_{kj}, & \text{se } y_{kj} > \widehat{y}_{kj} \\ 1, & \text{if } y_{kj} = \widehat{y}_{kj} \end{cases} \quad (6)$$

A soma em (5) é dividida em três grupos, dependendo da relação entre y_{kj} e $\widehat{y}_{kj}$:

$$Q_j = \sum_{k:\widehat{y}_{kj} > y_{kj}} (y_{kj} \equiv \widehat{y}_{kj}) + \\ + \sum_{k:\widehat{y}_{kj} = y_{kj}} (y_{kj} \equiv \widehat{y}_{kj}) + \sum_{k:\widehat{y}_{kj} < y_{kj}} (y_{kj} \equiv \widehat{y}_{kj}) \quad (7)$$

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

$$\Delta v_{kj} = \frac{\eta(t)}{N} \left[\frac{\partial Q_j}{\partial v_{kj}(t+1)} + \alpha \frac{\partial Q_j}{\partial v_{kj}(t)} \right] \quad (8)$$

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

$$\eta(t) = \frac{1}{\sqrt{t} + 10} \quad (9)$$

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

$$\frac{\partial Q_j}{\partial v_{kj}} = - \sum_{k:\widehat{y}_{kj} > y_{kj}} \frac{\partial \psi_{or}(\cdot)}{\partial v_{kj}} + \sum_{k:\widehat{y}_{kj} < y_{kj}} \frac{\partial \psi_{or}(\cdot)}{\partial v_{kj}} \quad (10)$$

$\psi_{or}(\cdot)$ é dado pela equação (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 $\widehat{y}_{kj}$ 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 $\widehat{y}_{kj}$ 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 demosntrada 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 $\delta = 0$, sendo que para cada unidade intermediária i os pesos são atualizados de acordo com a seguinte equação:

$$\Delta w_{ji} = \begin{cases} \alpha_1[1 - w_{ji}], & \text{if } \delta = 1 \\ -\alpha_2 w_{ji}, & \text{if } \delta = 0 \end{cases} \quad (11)$$

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, $\delta = 1 - \epsilon$, onde ϵ é a medida de desempenho

da rede. Observe que $0 \leq \delta \leq 1$ e qua 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:

$$\Delta w_{ji} = \delta\, \alpha_1\, [1 - w_{ji}] - (1 - \delta)\, \alpha_2\, w_{ji} \qquad (12)$$

onde $0 < \alpha_1 << \alpha_2 < 1$ são taxas de aprendizagem, $j = 1, \ldots, M,\ i = 1\ \ldots, n + M$ e $w_{jn+l} = r_{jl}, l = 1, \ldots, M$. O sinal de reforço é $\delta = (y_{kj} \equiv \widehat{y}_{kj})$ dado por(11).

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 (12) 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 depende do grau de associação entre a saída da rede e a saída desejada. As proporções lineares dos pesos mudam entre os casos extremos controlado por (11) o qual pode ser visto como uma particularidade de (12) 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 analisado 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 um modelo de série temporal ARIMA e com uma rede neural multicamadas com algoritmo de retropropagação, como pode ser visto em [15].

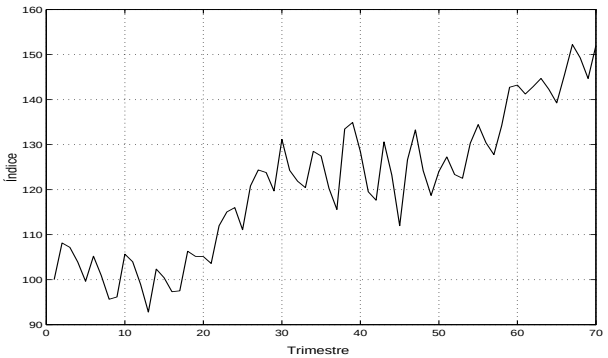

Figura 4: Produto interno bruto real.

Para ajuste do modelo somente 66 amostras da seqüência foram fornecidas, ou seja, foram fornecidos os dados a partir do primeiro semestre de 1980 até o segundo semestre de 1996. O objetivo foi prever os próximos 4 semestres (terceiro semestre de 1996 ao segundo semestre de 1997). A Figura 5 mostra as previsões 1 passo à frente e 4 passos à frente. A RFNR usada para modelar este processo tem uma entrada $x(t)$. Foi considerado uma partição não-uniforme com $N_1 = 16$ conjuntos fuzzy, o qual significa 16 neurônios na camada intermediária e 288 parâmetros para ajustar. As *t-normas* e *s-normas* escolhidas foram o produto algébrico e soma probabilística, respectivamente. As taxas de aprendizado usada para ajustar os pesos da camada intermediária foram $\alpha_1 = 0,01$ e $\alpha_2 = 0,001$. Os valores do índice de desempenho tornam-se estáveis, assumindo o valor $Q_j = 59,07$ após 1000 iterações.

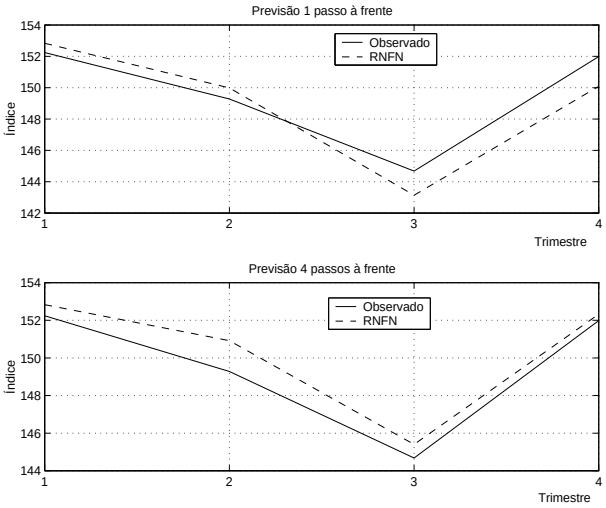

Figura 5: Predição da série produto interno bruto real.

Como mostra a Figura 4, a série apresenta uma pequena tendência de crescimento na economia brasileira no período. Da mesma forma há um padrão sazonal bem definido, que pode ser associado à períodos de aquecimento da economia no terceiro trimestre e de queda da atividade econômica no primeiro trimestre. 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 multicamadas (*MLP*) foi ajustada sendo que o vetor de entrada contêm 5 neurônios, ou seja, contêm o valor $x(t)$ da série e 4 neurônios binários de sazonalidade (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 (*REMQ*) e o erro médio absoluto (*EMA*).

Observe que, tanto para previsão 1 passo à frente como para previsão 4 passos à frente, o melhor modelo é o RNFN proposto, cujos erros são significativamente menores do que todos os demais. 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.

Tabela 1: Estatísticas das previsões.

Modelo	1 Passo à frente		4 Passos à frente	
	REMQ	EMA	REMQ	EMA
ARIMA	2,8700	2,1968	2, 5187	1,7936
MLP	2,2334	2,0106	1,8236	1,3000
RNFR	1,3073	1,1855	0,9611	0,8265

5. Conclusões

Neste artigo foi introduzido um novo algoritmo de aprendizado para estruturas de rede neurofuzzy recorrente. O algoritmo usa o índice de igualdade como medida de desempenho. Índice de igualdade é especialmente improtante devido suas propriedades refletirem a estrutura de cojunto 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 lógicas com operações definidas por *t-normas* e *s-normas*. A rede tem uma estrutura multicamada e uma relação dual com 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. Resultados de simulação mostram que a rede neural apresenta alto desempenho em termos de capacidade de aproximação, memória e tempo de processamento.

Agradecimentos

Esta pesquisa teve o suporte do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) processo $\sharp 300729/86 - 3$.

Referências

[1] R. J. Williams and 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*, Tese de Doutorado, Faculdade de Engenharia Elétrica e de Computação, Unicamp, 1996.

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

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

[5] C. H. Lee and C. C. Teng, "Identification and control of dynamic systems using recurrent fuzzy neural networks," *IEEE Trans. on Fuzzy Systems*, vol. 8, no. 4, pp. 3493–366, August 2000.

[6] R. Ballini, S. Soares, and F. Gomide, "A recorrent neurofuzzy network structure and learning procedure," in *Proc. 10th IEEE International Conference on Fuzzy Systems-FUZZ-IEEE'2001*, Melbourne, Australia, 2001, pp. 1408–1411.

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

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

[9] W. Caminhas, H. Tavares, F. Gomide, and 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 and 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.

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

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

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

[14] A. G. Barto and 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 Macroecômicas Agregadas no Ciclo de Negócios: Modeos Univariados de Séries de Tempo e Redes Neurais Artificiais*, Tese de Mestrado, Faculdade de Ciências Econômicas, UFRG, 1999.

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