10th Congresso Brasileiro de Inteligência Computacional (CBIC' 2011), 8 de Novembro de 2011, Fortaleza, Ceará Brasil

Redes Neurais Polinomiais Construtivas Baseadas em Inversão de Matrizes e Evolução Diferencial

Lorena G. N. Tablada e Mêuser J. S. Valença

Universidade de Pernambuco, Escola Politécnica de Pernambuco, Madalena, Recife, PE, 50720-001, Brazil
lorenatablada@gmail.com e meuserv@yahoo.com.br

Resumo – Um dos maiores problemas no que concerne às Redes Neurais Artificiais (RNAs) está relacionado à definição de sua arquitetura. Para atenuar esse problema, as Redes Neurais Polinomiais Construtivas (RNPCs) podem ser utilizadas. Entretanto, o emprego desta última acarreta em um aumento do custo computacional e no acréscimo de outros parâmetros, tais como o número de entradas em cada neurônio e o tipo de polinômio. A primeira desvantagem da utilização das RNPCs pode ter sua força reduzida por uma técnica de Inversão de Matrizes, enquanto que a segunda, por Evolução Diferencial, um tipo de Algoritmo Evolucionário. Neste trabalho, foi proposta uma combinação de técnicas para encontrar a melhor RNPC aplicada à tarefa de previsão de séries temporais. O método desenvolvido combina evolução diferencial, inversão de matrizes e RNPCs e busca simultaneamente pelos melhores valores dos parâmetros, da arquitetura e dos pesos da rede. Finalmente, foi possível observar que os erros obtidos com o método proposto foram bastante semelhantes ao de técnicas convencionais, porém com as vantagens de a arquitetura ser estruturada sem a interferência do usuário e com custo computacional bastante baixo.

Palavras-chave – Redes Neurais Polinomiais Construtivas, Inversão de Matrizes, Evolução Diferencial, séries temporais.

Abstract – One of the major problems regarding to Artificial Neural Networks (ANN) is related to the definition of its architecture. Self-Organizing Polynomial Neural Networks (SOPNN) can be used to alleviate this problem. However, it causes an increase in the computational cost and the addition of other parameters such as the number of entries in each neuron and the polynomial type. The first disadvantage of using SOPPNN can be mitigated by a technique of matrix inversion, while the second by Differential Evolution, a type of Evolutionary Algorithm. In this paper, it is proposed a combination of techniques in order to find the best SOPNN applied to the task of time series prediction. The developed method combines differential evolution, matrix inversion and SOPNN and simultaneously search for the best values of parameters, the network architecture and weights. Finally, it was noted that the errors obtained with the proposed method were very similar to those obtained with conventional techniques, but with the advantages of the architecture be structured without user interference and with very low computational cost.

Keywords – Self-Organizing Neural Networks, Matrix Inversion, Differential Evolution, time series.

1Introdução

Um dos maiores problemas no que concerne às Redes Neurais Artificiais (RNAs) está relacionado à definição de sua arquitetura. A quantidade de camadas escondidas e o número de neurônios em cada uma dessas camadas são parâmetros difíceis de serem configurados, principalmente através do método da tentativa e erro. Considerando-se que, para cada arquitetura configurada, serão realizadas algumas simulações (treinamento, validação cruzada e teste), é notório o alto custo computacional consequência do desconhecimento da arquitetura de melhor desempenho.

Com o intuito de atenuar o problema de definição de arquitetura de RNAs, as Redes Neurais Polinomiais Construtivas (RNPCs) podem ser utilizadas, pois essas são capazes de se auto-construir ao adicionar, testar e remover (quando não acrescentarem melhora à rede) neurônios e camadas à sua arquitetura. Entretanto, o emprego de RNPC acarreta em um aumento do custo computacional e no acréscimo de outros parâmetros, tais como o número de entradas em cada neurônio e o tipo de polinômio.

Enquanto que a primeira desvantagem da utilização das RNPCs pode ter sua força reduzida por uma técnica de Inversão de Matrizes, conhecida como Decomposição em Valores Singulares (SVD, acrônimo, do Inglês, de Single Value Decomposition), a segunda se enfraquecerá através do emprego da Evolução Diferencial, um tipo de Algoritmo Evolucionário que a mutação é uma das operações mais importantes para a convergência.

Assim, espera-se que a combinação desses algoritmos e técnicas torne mais brando um dos maiores problemas das RNAs, ou seja, a definição automática da sua arquitetura.

O restante do artigo foi dividido da seguinte forma: Nas seções 2, 3 e 4 são descritos o funcionamento das RNPCs, Inversão de Matrizes e Evolução Diferencial, respectivamente. Na seção seguinte, o experimento realizado para validar o sistema híbrido é explicado pormenorizadamente. Seguem-se, então, na seção 6, as conclusões obtidas a partir dos experimentos. E, por último, as referências utilizadas neste trabalho.

2 Redes Neurais Polinomiais Construtivas

Group Method of Data Handling (GMDH) foi desenvolvido no final dos anos 60 por Ivakhnenko[1-4] como um veículo para identificar relações não-lineares entre variáveis de entrada e saída. Desde a metade dos anos 70, tem sido usado para predição e modelagem de processos não-lineares complexos.

As principais características do GMDH são aauto-organização e a capacidade de prover uma seleção automatizada das variáveis de entrada essenciais sem usar informação a priori acerca das variáveis de entrada e saída. Porém, enquanto provê um procedimento de projeto sistemático útil, também possui algumas inconveniências. Primeiro, tende a gerar polinômios complexos para problemas relativamente simples. Segundo, devido à sua estrutura genérica limitada, o GMDH também tende a produzir redes demasiadamente complexas quando se trata de sistemas altamente não lineares. Terceiro, se existem menos do que três variáveis de entrada, o GMDH não gera uma estrutura muito versátil.

Para atenuar os problemas associados ao GMDH, Redes Neurais Polinomiais Construtivas (RNPCs) foram introduzidas por Oh [5-7], entre outros. Essas redes têm um alto nível de flexibilidade já que cada neurônio pode ter um número diferente de variáveis de entrada, diferentemente das RNAs comuns, onde o número de entradas é igual ao número de neurônios na camada imediatamente anterior. Além disso, as RNPCs podem explorar um diferente tipo de polinômio, como será visto adiante.

Formatado: Fonte: (Padrão) Times New Roman, 10 pt

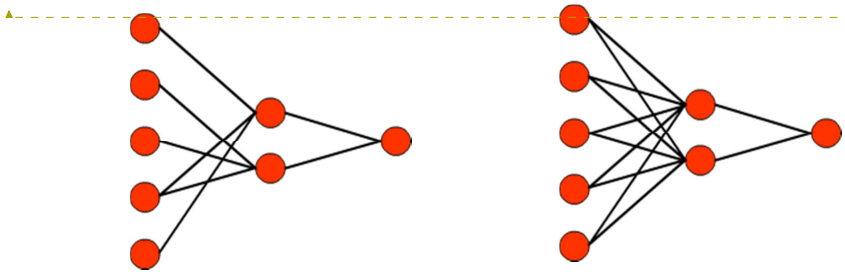

Figure 1 - Exemplos da representação de uma RNA simples (a) e uma RNPC (b).

O algoritmo da RNPC funciona da seguinte maneira:

1. **Primeiro passo:**Determinar as variáveis de entrada do sistema de acordo com a base de dados escolhida;
2. **Segundo passo:** Formar o conjunto de treinamento, validação cruzada e teste;
3. **Terceiro passo:** Escolher sua estrutura. Dois tipos de estrutura podem ser utilizados: a básica e a modificada:
   a) **Estrutura Básica:** O número de variáveis de entrada dos PDs é o mesmo em todas as camadas.
   b) **Estrutura Modificada:** O número de variáveis de entrada dos PDs varia de camada para camada.

   Quanto ao tipo de polinômio de cada neurônio, a estruturada rede pode ser dos seguintes casos:
   1. **Caso 1:**O tipo é o mesmo em todas as camadas da rede.
   2. **Caso 2:**O tipo na segunda ou próximas camadas difere do tipo na primeira camada.
4. **Quarto passo:** Determinar o número de variáveis de entrada e o tipo de polinômio formando um neurônio.Através da escolha da variável de entrada mais significante e da correta ordem dos polinômios, é possível construir a melhor descrição parcial em um neurônio polinomial. No estudo atual, as saídas dos neurônios individuais são expressos como equações de regressão de segunda ordem. Em particular, quando duas entradas são combinadas em cada neurônio, segundo a tabela 1, tem-se a seguinte relação:

$$y = A + BX_i + CX_j + DX_i^2 + EX_j^2 + FX_iX_j \quad (1)$$

na expressão acima, A, B, C, D, E e F são parâmetros, enquanto y é a saída do modelo. X_ieX_j denotam duas entradas. As saídas obtidas de cada um desses neurônios são então combinadas para se obter um polinômio de grau ainda maior.

Tabela 1. Diferentes tipos de polinômios em neurônios.

Tipo do Polinômio	Número de entradas	1	2	3
1		*Linear*	*Bilinear*	*Trilinear*
2		*Quadrático*	*Biquadrático*	*Triquadrático*
3		*Quadrático modificado*	*Biquadrático modificado*	*Triquadrático modificado*

Para duas entradas, essas são as saídas dos neurônios:

- Bilinear = $c_0 + c_1x_1 + c_2x_2$ (2)
- Biquadrático = $c_0 + c_1x_1 + c_2x_2 + c_3x_1^2 + c_4x_2^2 + c_5x_1x_2$ (3)
- BiquadráticoModificado = $c_0 + c_1x_1 + c_2x_2 + c_3x_1x_2$ (4)

5. **Quinto passo:** Estimar o erro de cada neurônio.
6. **Sexto passo:** Selecionar os neurônios com melhores capacidades preditivas.
7. **Sétimo passo:** Checar critério de parada.

Pode-se perceber que a utilização de RNPC acarretou no aumento do custo computacional, assim como no acréscimo de três parâmetros: o número de entradas em cada neurônio, o tipo de polinômio e a combinação de neurônios de entrada. Porém, o método da tentativa e erro é um trabalho pouco produtivo quando se trata da definição de tais parâmetros. Por essa razão, a busca por técnicas alternativas de otimização de tais parâmetros tem se tornado uma abordagem muito comum. Dessa maneira, novas pesquisas na área de sistemas híbridos como alternativa às técnicas convencionais estão sendo incentivadas.Os Sistemas Neurais Híbridos (SNHs) podem suprir as deficiências na celeridade de algoritmos de treinamento convencionais, assim como viabilizar a automatização de tarefas lentas e experimentais no ajuste paramétrico.

Em função de as RNPC não possuir camada escondida, a Inversão de Matrizes pode ser usada como uma técnica rápida e eficiente de treinamento, como será visto na terceira seção, ao passo que o algoritmo de Evolução Diferencial, descrito na quarta seção, dará suporte à definição dos novos parâmetros introduzidos pela RNPC.

3 Inversão de Matrizes

Dado o sistema matricial A.X = B, tem-se que solução:

$$X = A^{+}.B \quad (5)$$

onde A^{+} é a inversa de Moore-Penrose da matriz A.

A grande vantagem da inversa de Moore-Penrose é o fato de seu cálculo ser extremamente rápida, quando utilizada em conjunto com a Decomposição em Valores Singulares (SVD, acrônimo, do Inglês, de Single ValueDecomposition). Além disso, ela pode ser calculada tanto para matrizes quadradas quanto retangulares, as mais comumente encontradas em problemas reais. Formalmente, o SVD de uma matriz A real ou complexa é uma fatorização da forma:

$$A = U\Sigma V^T \quad (6)$$

Já a inversa de Moore-Penrose de A, representada por A^{+}, é dada por:

$$A^{+} = V\Sigma^{-1}U^T \quad (7)$$

Empregando inversão de matrizes ao problema proposto neste projeto, pode-se pensar em A como sendo a matriz de entradas da base de dados a ser utilizada; X, a matriz de pesos da rede; e B, as saídas da base de dados.Isso pode ser bem empregado porque, em uma RNA, suas saídas são calculadas através do produto de suas entradas pelos seus pesos. Dessa

maneira, aplicando a fórmula 5,sabemos que, calculando-se a inversa de Moore-Penrose da matriz de entradas a base e multiplicando o resultado pela matriz de saídas da base, a matriz de pesos da Rede Neural será obtida.

4 Evolução Diferencial

A Evolução Diferencial (ED) é um tipo de Algoritmo Evolucionário. Dessa maneira, é um modelo computacional baseado no processo de evolução das espécies, ou seja, na Teoria da Evolução de Darwin (1859), cujo principal mecanismo é a seleção natural. Esta última, por sua vez, favorece os indivíduos que tem melhor desempenho nesta competição, fazendo com que indivíduos mais aptos sobrevivam, gerem suas proles e tenham seus genes propagados ao longo das próximas gerações. Este processo pode favorecer a produção de indivíduos cada vez mais adaptados ao seu ecossistema.

Proposta originalmente por Storn e Price [8], a Evolução Diferencialé um método baseado em um conjunto de indivíduos (população) com busca direta para a otimização de funções contínuas não-lineares e não diferenciáveis. A ED precisa que poucos parâmetros sejam ajustados. É, portanto, um método simples, porém efetivo. Na ED, algumas operações são realizadas com os indivíduos sendo, segundo [9] a mutação a mais importante.

O algoritmo pode ser descrito da seguinte maneira:

1. **Iniciação:** O algoritmo é iniciado criando-se k subpopulações iniciais (P_1, P_2, ..., P_k), cada uma contendo a mesma quantidade de indivíduos (conjunto de genes), inicializados aleatoriamente e diferentes entre si;
2. **Oposição:** Para cada subpopulação A, criar outra subpopulação formada pelos indivíduos opostos de A. Sendo X um indivíduo de A, X_j, o gene j do indivíduo X e $\breve{X}_j$, o gene oposto de X_j:

$$\breve{X}_j = a_j + b_j - X_j \tag{8}$$

onde a_j é o maior valor do gene j e b_j, o menor valor do gene j dentre todos os indivíduos. Em seguida,selecionam-se os melhores indivíduos entre as duas subpopulações de acordo com suas aptidões.

3. **Mutação:** Para cada indivíduo base, um novo indivíduo é criado através da soma ponderada da diferença entre dois indivíduos com um terceiro indivíduo (todos da mesma subpopulação), dado por:

$$P_m = P_{base} + F \times (P_{r1} - P_{r2}) \tag{9}$$

4. **Cruzamento:** Após a mutação, cada indivíduo P_{base} é cruzado com seu respectivo vetor mutante P_m,gerando uma nova solução candidata P_c de acordo com a equação abaixo:

$$P_c = \begin{cases} P_m, \text{se } rand_j(0,1) < C_r \text{ ou } j = j_{rand} \\ P_{base} \end{cases} \tag{10}$$

onde o gene utilizado será o de P_m quando um número gerado aleatoriamente (para cada gene) for menor do que a taxa de cruzamento C_r, ou quando outro número gerado aleatoriamente (uma única vez por indivíduo) j_{rand} for igual a um parâmetro predefinido j (o que garantirá que P_c jamais será igual a P_{base}). Caso contrário o gene utilizado será o de P_{base}.

5. **Seleção:** A seleção é o processo que deve decidir qual vetor (P_{base} ou P_c) fará parte da próxima geração. É dada pela seguinte fórmula, sendo f a função custo:

$$P_s = \begin{cases} P_c, \text{se } f(P_c) \leq f(P_{base}) \\ P_{base} \end{cases} \tag{11}$$

6. Se a condição de parada não for satisfeita (que pode ser a validação cruzada), volta para o passo **2**.

Existe ainda outra operação que pode ser realizada com os indivíduos, a migração: Sabendo que cada subpopulação converge para uma solução de forma independente, em períodos regulares, a informação da busca é compartilhada através da migração de indivíduos de uma subpopulação para outra.

5 Experimentos e Resultados

Para a validação do modelo proposto, foram utilizadas cinco conhecidas bases de dados para problemas de previsão: Abalone,CaliforniaHousing, Delta Elevators, Machine CPU e Servo, a maioria disponível em http://archive.ics.uci.edu/ml.

A fim de mensurar o erro do modelo, foi empregada a métrica Raiz do Erro Médio Quadrático (REMQ), dada pela fórmula:

$$\text{REMQ} = \sqrt[2]{\text{EMQ}} \tag{12}$$

sendo EMQ:

$$\text{EMQ} = \frac{1}{\text{m.n}} \sum_{\text{e=1}}^{\text{e=m}} \sum_{\text{s=1}}^{\text{s=n}} (d_{e,s} - o_{e,s})^2 \tag{13}$$

Onde m é o número de exemplos, n é o número de saídas da rede, d é a saída desejada e o, a calculada.

No intuito de comparar os resultados obtidos no modelo proposto (com duas entradas e polinômio tipo dois) com as convencionais redes *Backpropagation* e as rápidas *Extreme Learning Machine* (*ELM*), foram utilizados dados obtidos em [10]. A tabela 2 sumariza tais resultados. Os valores apresentados para a rede RNPC são valores médios de 30 simulações.

Tabela 2. Comparação dos resultados obtidos com *Backpropagation*, ELM e RNPC.

Base de dados	***Backpropagation***	***ELM***	**RNPC**
Abalone	0,0785	*0,0803*	***0,0593***
California Housing	**0,1046**	*0,1217*	*0,1459*
Delta Elevators	0,0544	*0,0550*	***0,0432***
Machine CPU	0,0352	***0,0332***	*0,0349*
Servo	0,0794	***0,0707***	*0,0810*

Analisando-se os resultados das REMQs obtidos, é possível observar que os erros obtidos com a RNPC proposta neste estudo são bastante próximos, quando não menores do que os erros utilizando-se técnicas já consagradas de treinamento de uma rede neural.

6 Conclusões e Trabalhos Futuros

Este trabalho apresentou um modelo cujo foco foi a otimização da definição dos parâmetros de uma rede neural. O modelo proposto combina a qualidade das Redes Neurais Polinomiais Construtivas, Inversão de Matrizes e Evolução Diferencial para atenuar as desvantagens do primeiro modelo, como o custo computacional e o acréscimo de parâmetros.

A eficiência dos resultados foi medida através do cálculo da Raiz do Erro Médio Quadrático e esta foi, então, comparada aos resultados obtidos em outro trabalho. Assim, foi possível observar que os erros obtidos no modelo proposto neste estudo foram muito próximos aos obtidos utilizando-se outros algoritmos de treinamento já consagrados. Porém, a RNPC tem a vantagem de escolher, por si só, a arquitetura da rede que resulta no menor erro alcançável por ela. Além disso, o fato de várias arquiteturas serem testadas não acarreta em um aumento do tempo treinamento, fator mais desvantajoso da utilização do *backpropagation*. Em média, todos os erros foram obtidos em menos de 1 minuto.

Pode-se concluir este trabalho declarando-se que os resultados obtidos foram bastante satisfatórios. Esse desempenho das metodologias incentiva a continuação de mais pesquisas na mesma área. As extensões deste trabalho podem incluir:

- Transformação logarítmica no polinômio;

- Utilização de outras funções de ativação;
- Aplicar a rede à previsão de séries temporais com grande variabilidade, por exemplo, séries de velocidade do vento.

7 Referências

[1] A. G. Ivakhnenko, Polynomial theory of complex systems, **IEEE Trans. On Systems, Man and Cybernetics**, vol. SMC-1, pp. 364-378, 1971.

[2] A. G. Ivakhnenko and H. R. Madala, **Inductive Leraning Algorithms for Complex Systems Modeling**, CRC Press, London, 1994.

[3] A. G. Ivakhnenko and G. A. Ivakhnenko, **The review of problems solvable by algorithms of the group method of data handling (GMDH),** Pattern Recognition and Image Analysis, vol. 5, no. 4, pp. 527-535, 1995.

[4] A. G. Ivakhnenko, G. A. Ivakhnenko and J. A. Muller, **Self-organization of neural networks with active neurons**, Pattern Recognition and Image Analysis, vol. 4, no. 2, pp. 185-196, 1994.

[5] S.-K. Oh and W. Pedrycz, **The design of self-organizing polynomial neural networks**, Information Science, vol. 141, pp. 237-258, 2002.

[6] S.-K. Oh, W. Pedrycz and B.-J. Park,**Polynomial neural networks architecture: analysis and design**, Computers and Electrical Engineering, vol. 29, no. 6, pp. 703-725, 2003.

[7] H.-S. Park, B.-J. Park and S.-K. Oh, **Optimal design of self-organizing polynomial neural networks by means of genetic algorithms**, Journal of the Research Institute of Engineering Technology Development, vol. 22, pp. 111-121, 2002.

[8] R. Storn and K. Price, **Differential evolution – a simple efficient adaptive scheme for global optimization over continuous spaces**. Technical Report 95-012, Int. Compt. Sci. Inst., Berkeley, CA, 1995.

[9] A. M. F., Zarth, **Otimização Evolucionária Multimodal de Redes Neurais Artificiais com Evolução Diferencial**, Dissertação de Mestrado da Universidade Federal de Pernambuco, Março, 2010.

[10] G.-B. Huang, Q.-Y. Zhu, C.-K.Siew, **Extreme Learning Machine: Theory and applications**, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, 2005.