Software para a determinação da melhor arquitetura de uma Rede Neural Artificial utilizando Algoritmos Genéticos

Felipe D. Matos* **Marcelo F. G. Nogueira** **José C. Rocha**
Departamento de Ciências Biológicas, Engenharia Biotecnológica, FCL UNESP Assis
19806-900, Assis, São Paulo
E-mail: jcelso@assis.unesp.br

RESUMO

O correto desenvolvimento da arquitetura de uma Rede Neural Artificial (RNA) é peça chave na resolução eficiente de problemas [1], mas caso não seja corretamente determinada, a RNA pode não encontrar a melhor solução possível para o conjunto de dados apresentado, podendo ocorrer classificações errôneas ou problemas de overfitting quando exposta a dados de teste. Diversos fatores são responsáveis pela arquitetura, como a quantidade de camadas ocultas, o número de neurônios em cada camada, as funções de transferência entre os neurônios e a função de treinamento da RNA [2]. Entretanto, não há um método que possa determinar precisamente quais as melhores condições de arquitetura para a resolução de cada problema, existindo apenas algumas sugestões heurísticas que não necessariamente determinam sempre a melhor opção [2].

Buscando soluções para este dilema, desenvolvemos em ambiente *Matlab* um sistema baseado em algoritmos genéticos que busca a melhor configuração de arquitetura. Alguns trabalhos já exploram uma abordagem semelhante [3], apresentando resultados interessantes e demonstrando que algoritmos evolutivos atuam com eficiência na busca de uma melhor arquitetura. O funcionamento do algoritmo pode ser descrito nas seguintes etapas:

1) Criação da população inicial: Cada indivíduo representará uma arquitetura diferente, cujas informações estão contidas em seu "código genético" (também denominado *string*). Cada *string* possui as informações para quantidade de neurônios das camadas ocultas, Função de transferência para as camadas ocultas, Função de transferência para camada de saída, Função de treinamento a ser utilizada e quantidade de camadas ocultas. A quantidade total de indivíduos é determinada pelo usuário. As funções de transferência e treinamento são extraídas de uma lista pré-determinada (que também pode ser editada pelo usuário). Foram utilizadas como Funções de *transferência logsig, purelin, tansig, hardlim, tribas, radbas e satlin* e como Funções de treinamento *traingdx, trainscg e traingdm.* A quantidade de neurônios é aleatoriamente sorteada dentro de um intervalo estabelecido pelo usuário. A quantidade de camada ocultas é escolhida aleatoriamente, sendo no mínimo uma e no máximo três.

2) Teste e seleção da População: Uma vez criados os indivíduos (geradas as redes baseadas nas *strings*) toda a população é treinada e testada, utilizando-se o banco de dados para qual se deseja encontrar a melhor arquitetura. O usuário determina uma "faixa de corte", de modo que apenas melhores indivíduos (ou seja, os mais aptos) podem passar para a geração seguinte. O valor de corte corresponde à porcentagem da população com menor erro. Por exemplo, em uma população de 100 indivíduos, um valor de corte de 0,3 determina que apenas os 30 melhores (com menor erro) serão utilizados na geração seguinte, isto é, as 30 melhores arquiteturas da RNA serão utilizadas na geração seguinte.

3) Replicação e Repovoamento: Os indivíduos (arquiteturas) que passaram pelo corte formarão a população para a nova época de iteração do algoritmo, de forma que a quantidade total de indivíduos continue constante. Seguindo o exemplo dado, se em uma população de 100 indivíduos apenas 30 passarem para a próxima época, estes 30 serão os "progenitores" dos próximos 100 indivíduos. Dos selecionados, um par é escolhido ao acaso. Ocorre então a criação de uma nova *string*, que é uma combinação das *strings* dos progenitores. Cada característica (ou cada gene) é aleatoriamente escolhido entre um dos progenitores, até que a nova *string* esteja completa

*Bolsista Mestrado FAPESP (Processo nº 2012/20110-2)

(ou seja, um novo indivíduo para a próxima iteração). Também existe a possibilidade de se ocorrer uma mutação durante a criação da nova *string*, tornando possível a criação de variabilidade genética na população. A taxa de mutação (probabilidade de que possa ocorrer) também é determinada pelo usuário. Após, um novo par é selecionado aleatoriamente (dentre os vencedores da etapa anterior), e o processo se repete até que a população esteja novamente completa. Portanto, teremos uma nova população, que é combinação aleatória dos indivíduos vencedores da iteração anterior, acrescida da possibilidade de terem ocorrido mutações.

4) Obtenção da melhor arquitetura: O processo ocorre por quantas épocas forem determinadas pelo usuário, sempre selecionando os indivíduos mais aptos e recombinando suas *strings* para a próxima geração. Como combinações que causam um erro alto são eliminadas, a cada iteração o algoritmo se aproxima do conjunto de arquiteturas ótimas para a resolução do problema. Ao final de todas as gerações, o algoritmo apresenta a rede que obteve o menor erro para os dados de teste (dentre todas as possibilidades testadas). Nesse caso, não apenas são apresentadas as características de arquitetura da melhor rede, mas também a própria rede já corretamente treinada.

Para análise do algoritmo consideramos um banco de dados composto por 190 imagens de embriões bovinos, que devem ser classificados em 3 graus de qualidade. O sistema foi testado para encontrar a melhor arquitetura de rede utilizando-se, para cada imagem, 9 variáveis de entrada para a RNA. Foram utilizados 100 indivíduos, evoluindo por 100 gerações. A Figura 1 demonstra a evolução das populações:

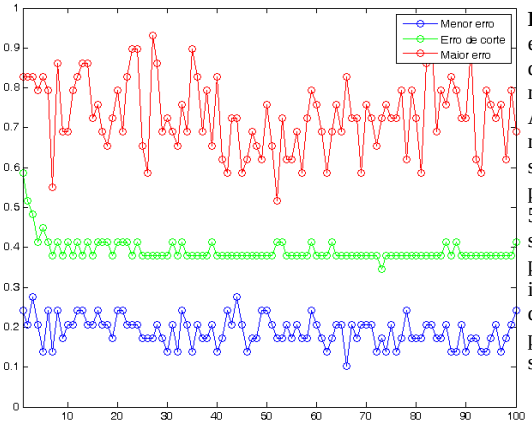

Figura 1 Gráfico demonstrando a evolução dos indivíduos através das gerações. A Linha azul mostra o indivíduo de menor erro. A linha verde o indivíduo com maior erro que ainda foi selecionado para compor a próxima geração (nesse caso os 50% melhores indivíduos são selecionados para gerar a próxima população). A linha vermelha indica o indivíduo com maior erro da época. O tempo total de processamento foi de 4169,08 segundos.

Os resultados mostraram que a melhor arquitetura definida pelo software foi de uma rede com uma camada oculta, com função de transferência *satlin* e 17 neurônios. A camada de saída possui função de transferência *purelin* e 3 neurônios, quantidade necessariamente igual ao número de saídas da classificação (Bom, Regular ou Ruim). Desta forma a RNA teve um acerto de 89,7% para os dados de teste. Este resultado comprova que a utilização de Algoritmos Genéticos se mostra eficiente na determinação da melhor arquitetura para as Redes Neurais Artificiais**.**

Palavras-chave: Redes Neurais Artificiais, Algoritmos Genéticos, Arquitetura

Referências

[1] I. Ramlall, Artificial intelligence: neural networks simplified. International Research Journal of Finance p. 1–20, 2009.

[2] S. Walczak e N. Cerpa, Heuristic principles for the design of artificial neural networks. Information and Software Technology, v. 41, n. 2, p. 107–117, 1999.

[3] X. Yao e Y. Liu, Towards designing artificial neural networks by evolution. Applied Mathematics and Computation, v. 91, n. 1, p. 83–90, 1998.

