Evoluindo os pesos de uma Rede Neural com Algoritmos Genéticos

Davi Azevedo Q. Santos[1], **Derik Evangelista R. Silva**[1], **Aurora Trinidad R. Pozo**[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19081 – 81531-980 – Curitiba – PR – Brasil

{daqsantos, dersilva, aurora}@inf.ufpr.br

Abstract. *This paper compares the training of a Multi Layer Perceptron using two approaches: a pure Genetic Algoritthm and a hybrid one using backpropagation. Various combinations of genetic operators are applied to the problem of classification.The results should show that the best combination can meet or exceed the results obtained with the classic Backpropagation algorithm.*

Resumo. *Este artigo compara o treinamento de um Multi Layer Perceptron usando duas abordagens: um Algoritmo genético puro e um híbrido usando backpropagation. Várias combinações de operadores genéticos são aplicadas para o problema de classificação. Os resultados devem mostrar que a melhor combinação consegue atingir ou ultrapassar os resultados obtidos com o algoritmo clássico de Back propagation.*

1. Introdução

As Redes Neurais Aritificiais são uma das formas mais populares e efetivas para construção de sistemas de aprendizagem [Russell and Norvig 2010], e são utilizadas nas mais diversas áreas, como, por exemplo, análise financeira [Raghupathi et al. 1991], medicina [Abbass 2002] e meteorologia [Raj 2008]. Representam uma flexível técnica heurística para reconhecimento de padrões [Duda et al. 2001], além de possuirem a habilidade de realizar computação distribuída, tolerar entradas ruidosas e, é claro, aprender [Russell and Norvig 2010].

A rede neural mais comumente utilizada é a Multi-layer Perceptron, na qual normalmente se utiliza o algoritmo *back propagation* [Rumelhart et al. 1988] para o treinamento dos pesos das conexões [Liu et al. 2004]. Apesar da popularidade, o *back propagation* tem algumas desvantagens. Por utilizar a gradiente descendente, a performance do treinamento está diretamente relacionada com a superfície de erro do problema, que pode conter mínimos locais, fazendo com que o gradiente descendente não encontre o mínimo global [Rumelhart et al. 1988].

Os algoritmos genéticos (AG) [Goldberg 1989], por sua vez, tem tido grande sucesso em problemas de busca e de otimização, principalmente em espaços de busca grandes, complexos e pouco conhecidos, onde os métodos de buscas convencionas (enumerativos, heurísticos, ...) não são apropriados, oferecendo um método válido para problemas que necessitam de uma busca eficiente e efetiva [Herrera et al. 1998].

O objetivo deste trabalho é treinar uma rede neural artificial utilizando algorítmo genético. [Liu et al. 2004] realiza o treinamento utilizando, além do AG, o *back propagation* para refinar a solução encontrada. Já [Montana and Davis 1989] utiliza um conjunto

variado de operadores genéticos de mutação e cruzamento. Neste trabalho, implementaremos uma versão híbrida dos métodos de treinamentos propostos por [Liu et al. 2004] e [Montana and Davis 1989] e compararemos os resultados do treinamento realizado com o AG puro contra o AG com *back propagation*.

Este artigo está organizado da seguinte forma: no capítulo 2 são apresentados os conceitos de Redes Neurais Artificiais e Algoritmos Genéticos. No capítulo 3 são apresentados os detalhes de implementação, operadores genéticos e as bases de dados utilizados. O capítulo 4 expõe os resultados; e o capítulo 5, as conclusões.

2. Metodologia

2.1. Redes Neurais

Segundo [Kasabov 1996] uma Rede Neural Artifical é um modelo computacional biologicamente inspirado no cérebro humano que consiste de elementos de processamento (neurônios) e conexões entre eles (pesos) que representam a memória do sistema. O primeiro modelo de neurônio artificial foi proposto por McCulloch e Pitts em 1943.

Uma rede neural é definida por quatro parâmetros:

1. Tipo de neurônio: determina como são tratados os valores de entradas e calculados de saída de um neurônio;
2. Arquitetura conexionista: determina como os neurônios se conectam entre si;
3. Algoritmo de aprendizado: determina como o conhecimento é armazenado na rede;
4. Algoritmo de Recall: determina como o conhecimento da rede é recuperado.

Os vários modelos de redes neurais artificais podem ser descritos em termos destes parâmetros. O modelo mais comum é o Multi Layer Perceptron (MLP), que consiste de uma arquitetura totalmente conectada entre as camadas, possui no mínimo três camadas (entrada, oculta e saída). Além disso, cada neurônio tem um entrada fixa (bias), e a função de ativação mais usada é a sigmóide.

Neste trabalho utilizaremos o MLP para o problema de classificação. Segundo [Kasabov 1996] O problema de classificacão é associar um objeto a um grupo ou classe de objetos já existentes.

O MLP tornou-se bastante utilizado com surgimento do algoritmo Backprogation como algoritmo de aprendizado. Contudo, neste trabalho, para realizar o treinamento da rede neural, será implementado duas abordagens: uma utilizado um Algoritmo Genético puro e a outra um Algoritmo Genético híbrido, o qual introduz a execução do backpropagation logo após a aplicação dos operadores de mutação. Todavia, por conta do custo, apenas um pequeno número de iterações do backprogation será utilizado.

2.2. Algoritmos Genéticos

O Algoritmo Genético (AG), geralmente referenciado como *algoritmos genéticos*, foi desenvolvido por John Holland na Universidade de Michigan, em 1975, em um dos livros mais famosos da área, *Adaptation in Natural and Artificial Systems*, publicado pela editora *University of Michigan Press* [Luke 2009b].

Os AGs são algoritmos de busca heurística que imitam artificialmente o processo de evolução, da Teoria de Evolução das Espécies, de Darwin. São muito similares a outros algoritmos de busca local, como o Subida de Encosta (*Hill-climbing*), diferindo apenas na quantidade de soluções mantidas a cada iteração e no processo de geração de novas soluções.

De acordo com [Montana and Davis 1989], os AGs necessitam de cinco componentes:

C1 - Uma maneira de codificar uma solução em um indivíduo.
C2 - Uma função de avaliação que retorna um índice de qualidade para cada individuo da população.
C3 - Um procedimento de inicialização da população.
C4 - Operadores que podem ser aplicados nos indivíduos pais no processo de reprodução, alterado a composição genética dos novos indivíduos gerados. Neste componente incluem-se os operadores de mutação, de cruzamento, seleção e outros específicos do domínio.
C5 - Uma configuração de parâmetros do algoritmo, os operadores, etc.

Com estes cinco componentes, ainda de acordo com [Montana and Davis 1989], o AG funciona seguindo os seguintes passos:

1. A população é iniciada usando-se o procedimento C3. O resultado é um conjunto de indivíduos, de acordo com C1.
2. Cada indivíduo é avaliado, usando a função definida em C2.
3. A população se reproduz até que um critério de parada seja atingido. A reprodução é realizada seguindo-se os seguintes passos:
   (a) Um ou mais indivíduis são escolhidos para a reprodução. A seleção é estocástica, mas os pais melhores avaliados são favorecidos na escolha. Os parametros em C5 podem influenciar neste processo de seleção.
   (b) Os operadores de C4 são aplicados aos pais para geração dos filhos. Os parametros em C5 ajudam a determinar quais operadores serão usados.
   (c) Os filhos são então avaliados e inseridos de volta na população. Em algumas versões de AG, toda a população é substituida. Em outras, apenas um subconjunto é substituido.

A cada iteração, os indivíduos melhor avaliados tem mais chances de serem escolhidos para reprodução, fazendo com que, em teoria, sejam gerados melhores indivíduos a cada geração. Ao término do algoritmo, senão a ótima, a tendência é ter-se uma solução muito próxima a ela.

3. Implementação

A linguagem Java foi utilizada para implementar os algoritmos. A rede neural foi codificada como um vetor de números em ponto flutuante, também conhecido como *real-coded* [Liu et al. 2004]. A figura 1 ilustra esse processo, que foi baseado em [Montana and Davis 1989]. Desta forma, é possível determinar rapidamente quais são os pesos de entrada e saída de cada neurônio e assim facilitar a implementação dos operadores genéticos. O número de camadas da rede é fixo e consiste em uma camada de entrada, uma camada oculta e uma camada de saída.

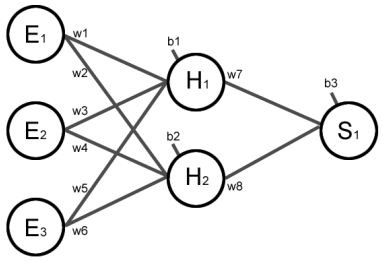
As bases de dados utilizadas foram retiradas de [Frank and Asuncion 2010]. Para o experimento foram utilizadas as seguintes bases:

1. *Breast Cancer Wisconsin (Original) Data Set*, doravante denominada *Câncer*;
2. *Pima Indians Diabetes Data Set*;
3. *Glass Identification Data Set*;
4. Statlog (Heart) Data Set, doravante denominada *Heart*;
5. *Iris Data Set*.

Na tabela 1, temos o número de neurônios de cada camada para cada base de dados a ser testada.

Base	Atributos (Entradas)	Classes (Saídas)	Ocultos
Cancer	9	2	5
Pima Indians Diabetes	8	2	10
Glass Identification	9	7	10
Heart	13	2	5
Iris	4	3	10

Tabela 1. Número de neurônios de cada camada



Codificação da rede: [w1, w4, w5, b1, w2, w4, w6, b2, w7, w8, b3]

Figura 1. Codificacão da rede

O Algoritmo Genético foi implementado conforme [Luke 2009a]. O *fitness* de cada indivíduo é dado pelo Erro Quadrado Médio (EQM), ou seja, um indivíduo mais apto é aquele que possui o menor valor. O EQM é calculado de acordo com [Liu et al. 2004].

A seguir temos operadores genéticos utilizados. Seus parâmetros, quando possivel, foram os mesmos das respectivas referências.

1. Operadores de Mutação descritos em [Montana and Davis 1989] e [Liu et al. 2004] são:
   (a) Biased Mutate Weights: O valor de um gene pode ser substituído por um outro qualquer da distribuição de probabilidade inicial (distribuição normal com média 0 e desvio padrão 5).
   (b) Unbiased Mutate Weights: Um valor da distribuição de probabilidade inicial é acrescido a um gene.
   (c) Mutate Nodes: Este operador seleciona n neurônios das camadas oculta e de saída e adiciona um valor da distribuição de probabilidade inicial a cada peso de entrada dos neurônios. Nos nossos o valor de n foi 2.

(d) Mutate Weakest Nodes: Este operador seleciona o neurônio mais fraco (aquele que menos contribui para o saída da rede) e aplica uma mutação *unbiased* ou *biased* em cada peso do neurônio mais fraco.
(e) SinglePointRandom: Cada gene do cromossomo é substituído por um valor aleatório entre [-50, 50].
(f) NonUniform: implementado conforme [Michalewicz 1994]. Na nossa implementacão o parâmetro b é 4.

2. Operadores de Crossover, segundo [Montana and Davis 1989] e [Liu et al. 2004]:
   (a) Crossover Weights: O valor gene de um indivíduo filho é escolhido pela seleção aleatória do mesmo gene de um dos pais.
   (b) Crossover Nodes: Para cada neurônio de um dos pais escolhidos aleatoriamente os pesos associados são passados diretamente para o filho.
   (c) Crossover Features: Para cada neurônio da rede do primeiro pai, o algoritmo reorganiza o segundo pai de forma que os neurônios que desempenham o mesmo papel (isto é, possum a mesma saída para uma determinada entrada) fiquem na posição, assim formando um pai intermediário. Depois disso é aplicado o operador Crossover-Nodes entre o primeiro pai e o pai intermediário.
   (d) Line Recombination: A implementação foi um hibrido entre [Liu et al. 2004] e [Luke 2009a] onde o parâmetro é dado por uma distribuição normal com média 0 e desvio padrão 5.

O operador de seleção utilizado, diferente de [Liu et al. 2004] e [Montana and Davis 1989], foi o torneio.

Além disso foi implementado, para efeitos de comparação, um AG híbrido com backpropagation. A implementacão foi confome [Freeman and Skapura 1991]. Contudo, diferente de [Liu et al. 2004], neste experimento são executadas cinco iterações do backpropagation logo após a aplicacão dos operadores de mutação, com a finalidade de refinar as soluções

Os parâmetros utilizados para o Algoritmo Genético foram os seguintes:

1. Tamanho da População: 50 e 100 indivíduos;
2. Número de Gerações: 500;
3. Tamanho do torneio: 2 e 4;
4. Número de indivíduos da elite: 10;
5. Probabilidade de Crossover: 100%;
6. Probabilidade de mutação: 10%.

Para a análise dos resultados foi utilizado a média da taxa de acerto das 10 execuções para cada combinação de parâmetros (população, tamanho do torneio, operador de mutação e crossover). O conjunto de dados foi separado de acordo com a tabela 2.

4. Resultados obtidos

Foram realizadas 10 execuções para cada combinação possível de:

1. Parâmetros do Algoritmo Genético (explicitados na seção 3);
2. Operadores de mutação;

Base	Treinamento	Teste	Total
Cancer	525	174	699
Pima Indians Diabetes	510	258	768
Glass Identification	164	50	214
Heart	200	70	270
Iris	110	40	150

Tabela 2. Separação dos conjuntos de dados

3. Operadores de cruzamento;

Para os operadores de mutação e cruzamento, considerou-se também um operador extra, que seleciona aleatoriamente qualquer um dos operadores descritos. Nas figuras a seguir o eixo horizontal representa as gerações e o vertical o fitness médio de cada combinação de parâmetros.

4.1. Influência dos parâmetros do AG

De uma maneira geral, um tamanho de torneio maior mostrou ter um impacto positivo na média dos valores de fitness. Entretanto, a média piora em alguns casos ou, nos casos em que há melhora, ela é muito pequena, como podemos ver nas figuras 2 e 3.

Analisando as figuras 4 e 5 podemos ver que um maior tamanho da população apresenta uma influência bem mais significativa que o tamanho do torneio em praticamente todas as execuções. Entretanto, ainda assim, a relação de consumo de recursos computacionais, associado ao maior tempo para execução e o ganho real de uma população maior pode não ser vantajoso.

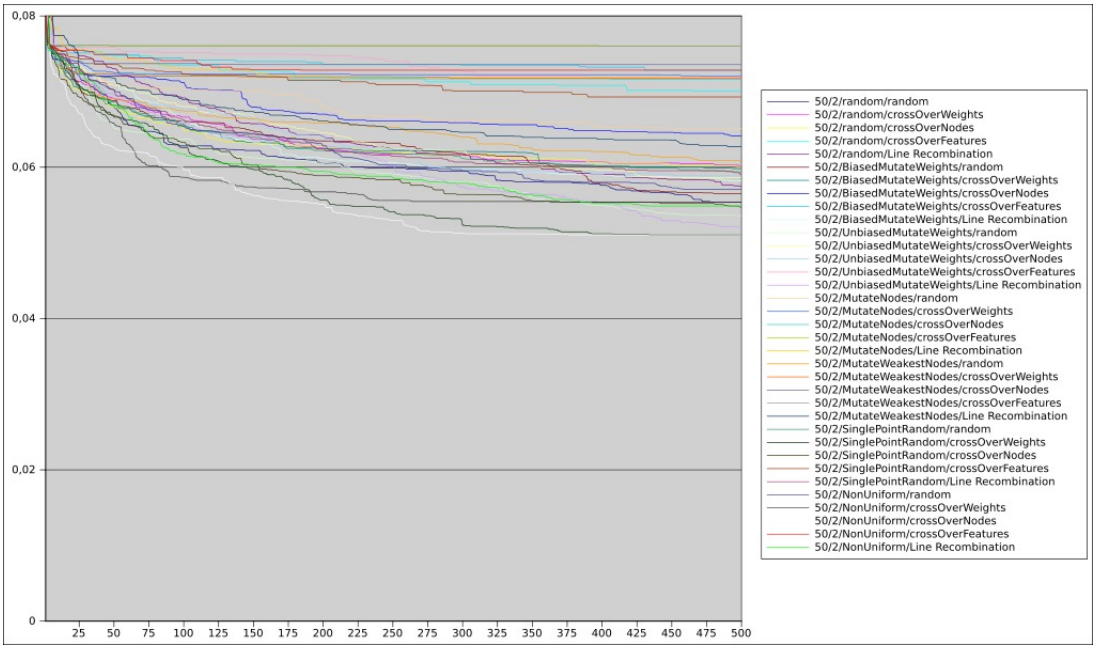


Figura 2. Fitness - Glass; População 50, torneio 2;

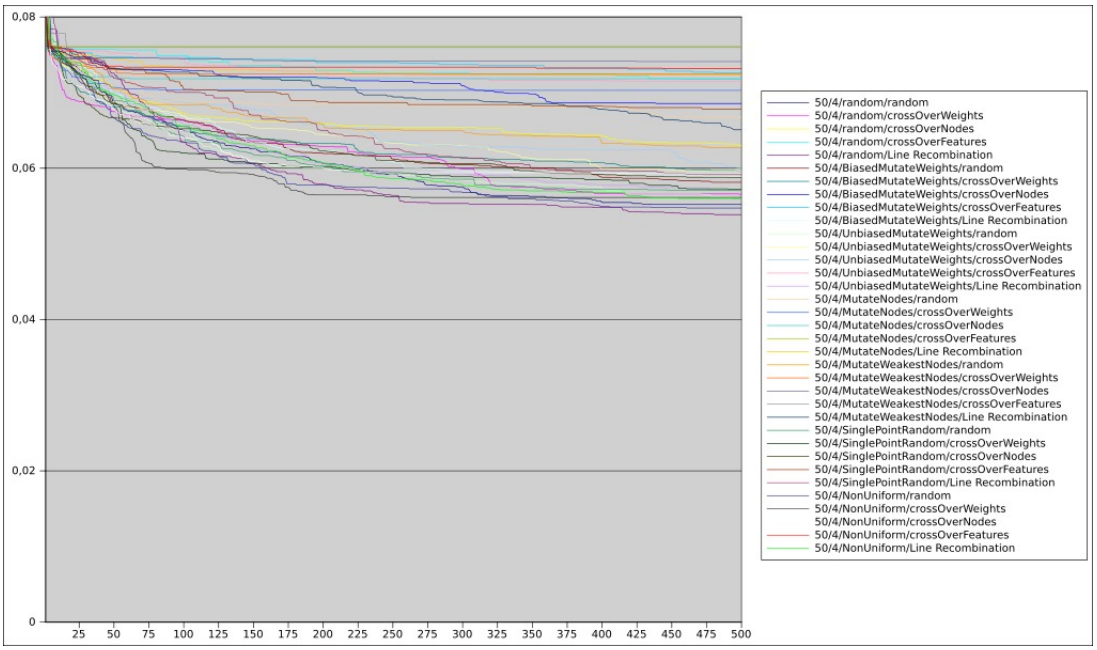


Figura 3. Fitness - Glass; População 50, torneio 4;

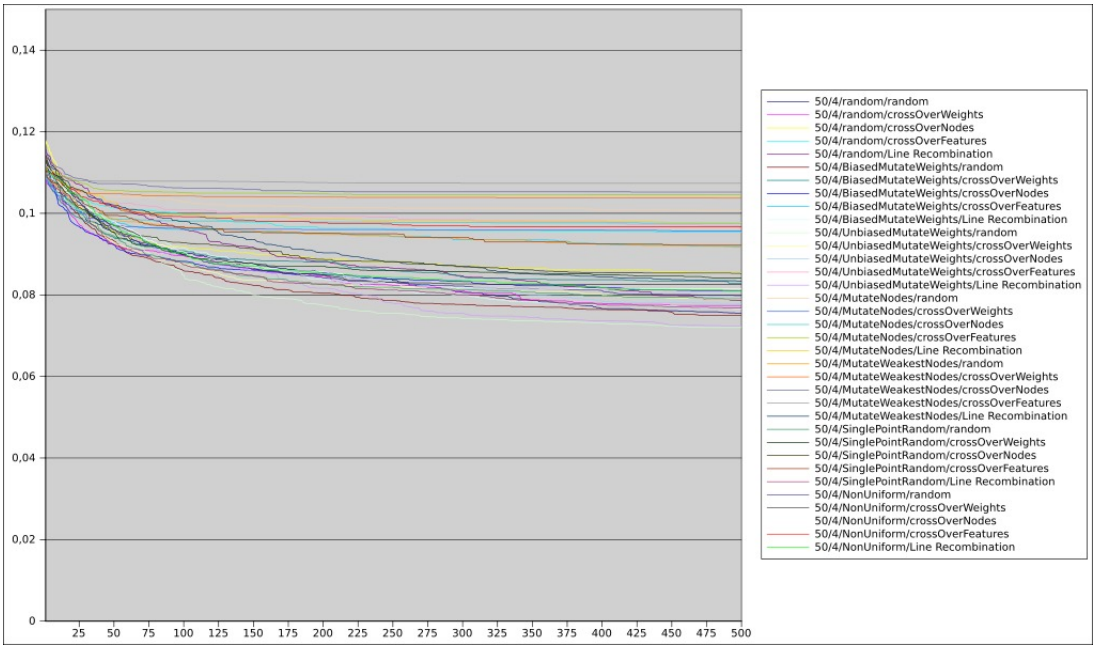


Figura 4. Fitness - Diabetes; População 50, torneio 4;

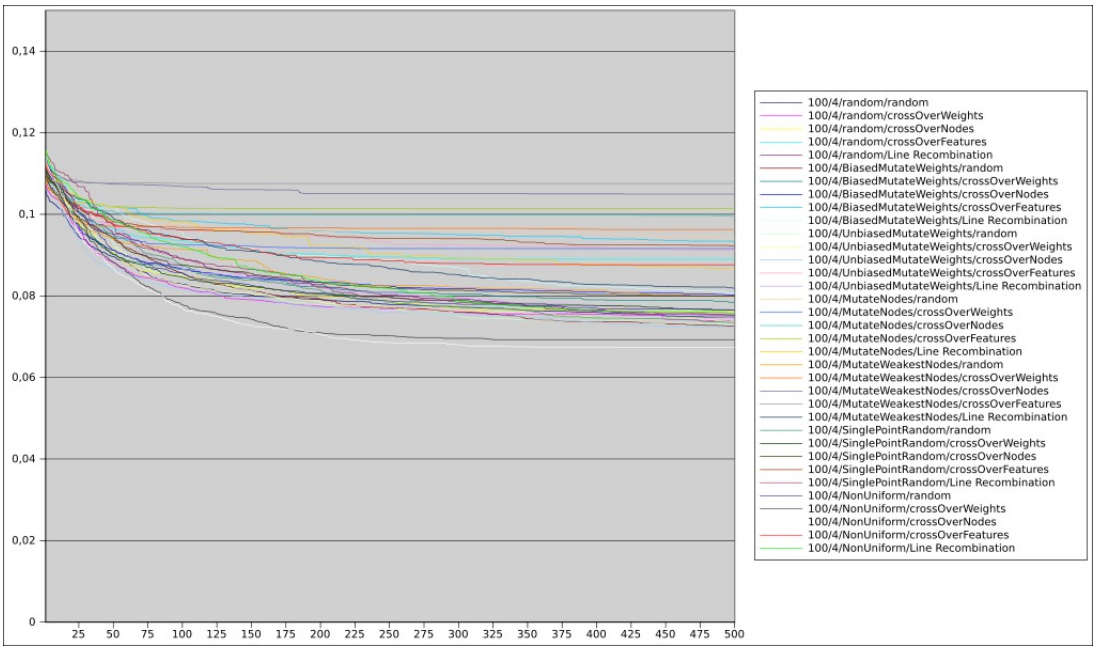


Figura 5. Fitness - Diabetes; População 100, torneio 4;

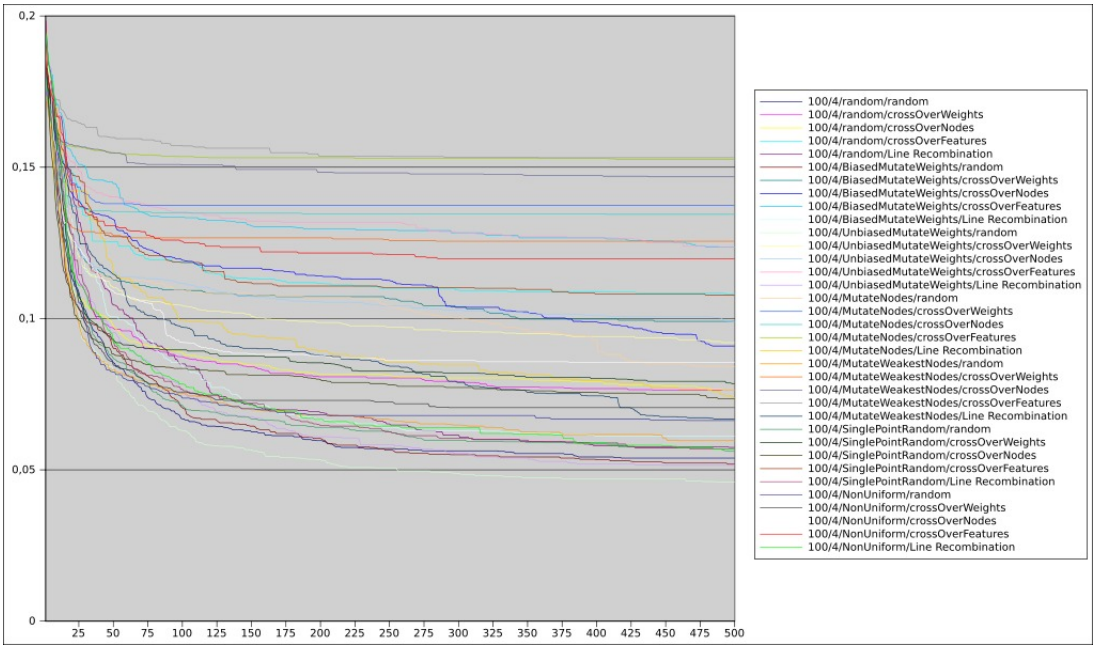


Figura 6. Fitness - Heart; População 100, torneio 4;

4.2. Influência dos Operadores Genéticos

Diferentemente dos parâmetros em 4.1, a escolha certa dos operadores genéticos tem um impacto muito grande. Podemos observar facilmente este impacto analisando a figura 6: a diferença entre a pior combinação de operadores, que termina com a média de fitness um pouco maior que 0.15 e a melhor combinação, que termina com um valor de fitness menor que 0.05.

Também é interessante notar que em algumas casos, como o mostrado na figura 4, muitas combinações de operadores mantem os valores de fitness muito próximos entre si e, como visto na figura 5, há uma tendência a, em algum ponto, estabilizar, passando muitas gerações sem uma melhora sensível no valor. Estas características encontradas podem ser, em partes, remediadas aplicando-se, tal qual proposto por [Montana and Davis 1989], uma espécie de torneio com os operadores, dando preferência para aqueles que estão produzindo um maior impacto na rede.

4.3. Taxas de Acerto

Ao término das execuções, conseguimos analisar qual a taxa de acerto média da rede neural com seus pesos evoluidos através de um Algoritmo Genético.

Na tabelas 3 e 4, podemos visualizar as melhores taxas de acerto para o conjunto de testes com o AG puro e com o AG com back propagation, assim como qual combinação de parâmetros e operadores maximizou o acerto. É interessante notar que o operador de cruzamento Line Recombination (dos cinco possíveis) foi que o conseguiu o melhor resultado na maioria das bases testadas. Já os operadores de mutação são mais heterogêneos. Podemos concluir, então, que o cruzamento tem um efeito maior na saída da rede que a mutação. Entretanto, se considerarmos que a mutação ocorre apenas em 10% das vezes, ao passo que o cruzamento ocorre sempre, este comportamento é esperado.

Também podemos perceber que, em alguns casos, a rede se comportou melhor com uma população e torneio menores, embora na maior parte das execuções uma grande população e um maior tamanho de torneio tenham superado as outras combinações.

Ao compararmos as médias das duas abordagens (AG puro e AG híbrido), percebemos que, em geral, o back propagation melhorou o desempenho dos testes. Todavia, apenas na base Glass, o AG puro foi um pouco melhor.

Na tabela 5 temos os valores de acerto esperado para cada base. Comparando esta tabela com o resultado obtido, à exceção da base Câncer e Iris, o desempenho da rede foi bem abaixo do esperado, principalmente na bases Glass. Analisando as figura 3, percebemos que o fitness para esta base converge rapidamente para um mínimo local e estaciona, não conseguindo sair deste mínimo. Por esse motivo, os valores encontrados para estas bases são relativamentes baixos. Outro ponto a ser notado, é que a base que possui o maior desvio padrão. Isso mostra que o AG tende a ficar preso em ótimos locais.

5. Considerações Finais

Baseado na análise dos resultados obtidos, podemos notar que, apesar do algoritmo conseguir uma rápida queda nos valores de fitness em poucas gerações, o que aumenta as taxas de acerto da rede, a evolução dos pesos da rede sofre com o problema recorrente aos AGs, de otimização fina. Ao chegar em um certo valor de fitness, por mais que se alterem

	Parâmetros				Taxa de acerto	
Base	Pop	Tor	Mutação	Cruzamento	Média	Desvio padrão
Câncer	50	2	Aleatório	Line Recombination	**0.961142**	0.000908
Diabetes	100	2	Unbiased Mutate Weights	Line Recombination	**0.663953**	0.002083
Glass	100	4	NonUniform	Aleatório	**0.414**	0.029725
Heart	100	4	Single Point Random	Line Recombination	**0.784285**	0.005421
Iris	100	4	Unbiased Mutate Weights	Crossover Weights	**0.96**	0.018973

Tabela 3. Melhores taxas de acerto para o AG puro

	Parâmetros				Taxa de acerto	
Base	Pop	Tor	Mutação	Cruzamento	Média	Desvio padrão
Câncer	50	4	Biased Mutate Weights	Aleatório	**0.963218**	0.00109
Diabetes	50	2	Unbiased Mutate Weights	Line Recombination	**0.707751**	0.017649
Glass	50	4	Aleatório	Line Recombination	**0.306**	0.020871
Heart	50	4	Single Point Random	Line Recombination	**0.798571**	0.004969
Iris	50	4	Unbiased Mutate Weights	Crossover Nodes	**0.9775**	0.008696

Tabela 4. Melhores taxas de acerto para o AG com back propagation

Base	Taxa de acerto média	Desvio padrão
Câncer	0.55	0.116563
Diabetes	0.46	0.195496
Glass	0.29	0.093893
Heart	0.52	0,079910
Iris	0.45	0.108857

Tabela 5. Taxas de acerto do back propagation

os códigos genéticos dos filhos, é muito difícil continuar a melhorar a rede, causando uma estagnação dos valores. É provavel que, mesmo que aumentemos o número de gerações, a melhora da rede seja ínfima. O mesmo acontece com o AG híbrido. Apesar de encontrar na maioria das vezes uma solução melhor o ganho geral é pequeno.

Para evitar isto, se faz necessário a criação de novos (e melhores) operadores genéticos, que consigam impactar a rede de forma que os pesos continuem a evoluir. Além disso, desenvolver um mecanismo de evolução dos operadores, conforme descrito por [Montana and Davis 1989], de forma que os operadores sejam escolhidos de acordo com o impacto que este teve anteriormente na rede, pode melhorar ainda mais a evolução

dos pesos. Isto pode ser feito em trabalhos futuros.

Referências

Abbass, H. A. (2002). An evolutionary artificial neural networks approach for breast cancer diagnosis. *Artif. Intell. Med.*, 25(3):265–281.

Duda, R., Hart, P., and Stork, D. (2001). *Pattern classification*. Pattern Classification and Scene Analysis: Pattern Classification. Wiley.

Frank, A. and Asuncion, A. (2010). UCI machine learning repository.

Freeman, J. A. and Skapura, D. M. (1991). *Neural networks: algorithms, applications, and programming techniques*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

Herrera, F., Lozano, M., and Verdegay, J. L. (1998). Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artif. Intell. Rev.*, 12(4):265–319.

Kasabov, N. K. (1996). *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*. MIT Press, Cambridge, MA, USA, 1st edition.

Liu, Z., Liu, A., Wang, C., and Niu, Z. (2004). Evolving neural network using real coded genetic algorithm (ga) for multispectral image classification. *Future Gener. Comput. Syst.*, 20(7):1119–1129.

Luke, S. (2009a). *Essentials of Metaheuristics*. Lulu, 15th edition. Disponível em: <http://cs.gmu.edu/∼sean/book/metaheuristics/>.

Luke, S. (2009b). Populational methods. In Luke, S., editor, *Essentials of Metaheuristics*, pages 29–56. Lulu.

Michalewicz, Z. (1994). *Genetic algorithms + data structures = evolution programs (2nd, extended ed.)*. Springer-Verlag New York, Inc., New York, NY, USA.

Montana, D. J. and Davis, L. (1989). Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th international joint conference on Artificial intelligence - Volume 1*, IJCAI'89, pages 762–767, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Raghupathi, W., Schkade, L., and Raju, B. (1991). A neural network application for bankruptcy prediction. In *System Sciences, 1991. Proceedings of the Twenty-Fourth Annual Hawaii International Conference on*, volume iv, pages 147 –155 vol.4.

Raj, V. J. (2008). Better performance of neural networks using functional graph for weather forecasting. In *Proceedings of the 12th WSEAS international conference on Computers*, ICCOMP'08, pages 826–831, Stevens Point, Wisconsin, USA. World Scientific and Engineering Academy and Society (WSEAS).

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Neurocomputing: foundations of research. chapter Learning representations by back-propagating errors, pages 696–699. MIT Press, Cambridge, MA, USA.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall.