X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará

ALGORITMO BASEADO EM CARDUMES DE PEIXES MODIFICADO PARA A OTIMIZAÇÃO DE EXTREME LEARNING MACHINES

João Fausto Lorenzato de Oliveira, Teresa B. Ludermir
Centro de Informática – Universidade Federal de Pernambuco (UFPE)
{jflo, tbl}@cin.ufpe.br

Resumo –Redes Neurais têm sido bastante aplicadas em vários problemas de classificação de padrões do mundo real. Durante o treinamento, toda rede neural pode sofrer perda de generalização causada por um super ajustamento da rede, tornando o processo de aprendizado altamente tendencioso. Neste trabalho utilizamos o *Extreme Learning Machine*, que é um algoritmo para o treinamento de redes neurais com apenas uma camada escondida, e propomos um novo algoritmo baseado em enxames para otimizar os pesos da rede e melhorar a sua capacidade de generalização. O algoritmo é composto pelo algoritmo de otimização baseado em cardumes de peixes, e algumas características de evolução diferencial (mutação e cruzamento) para melhorar a qualidade das soluções durante o processo de busca. O resultado das simulações demonstrou boa capacidade de generalização de todos os indivíduos e boa acurácia no conjunto de testes.

Palavras-chave – Redes Neurais, Algoritmos Evolucionários, Otimização.

Abstract – Neural networks have been largely applied into many real world pattern classification problems. During the training phase, every neural network can suffer from generalization loss caused by overfitting, thereby the process of learning is highly biased. For this work we use Extreme Learning Machine which is an algorithm for training single hidden layer neural networks, and propose a novel swarm-based method for optimizing its weights and improving generalization performance. The algorithm presents the basic *Artificial Fish Swarm Algorithm* and some features from *Differential Evolution* (Crossover and Mutation) to improve the quality of the solutions during the search process. The results of the simulations demonstrated good generalization capacity from all individuals and high accuracy on the test set.

Keywords – Neural Networks, Evolutionary Algorithms, Optimization.

1. INTRODUÇÃO

Uma das tarefas exploradas no campo da inteligência artificial é a classificação de padrões. Existem vários algoritmos que podem ser aplicados para este propósito, porém neste trabalho serão utilizadas redes neurais artificiais (RNAs). Muitos algoritmos clássicos de treinamento de RNAs são baseados em gradiente descendente, como por exemplo o *backpropagation* e o *Levenberg Marquardt* [1]. Nos últimos anos, esses algoritmos foram utilizados em diversos problemas do mundo real, incluindo tarefas como otimizações e buscas. No entanto, os algoritmos baseados em gradiente descendente são lentos e facilmente ficam presos em mínimos locais [1].

Existe a possibilidade encontrar um conjunto de pesos e bias que potencializem a capacidade de aprendizado da rede, e melhorem sua capacidade de generalização. Encontrar os grupos de pesos e bias ótimos não é uma tarefa simples, e necessita de algoritmos de busca bem fundamentados para sua realização. Portanto, este problema é caracterizado por um conjunto de variáveis (pesos e bias), uma função de custo (erro médio quadrático) e as restrições de norma dos pesos que serão abordadas mais à frente.

Uma abordagem para encontrar uma boa solução é utilizar algoritmos de otimização como *Particle Swarm optimization* (PSO) [2], Algritmos Genéticos (GA) [2] e *Artificial Fish Swarm Algorithm* (AFSA) [3], que possuem uma característica em comum, que é encontrar a melhor solução em um espaço de soluções. Para avaliar cada indivíduo, utiliza-se uma função de custo, que mede a qualidade da solução encontrada, e em seguida essa solução é comparada com as demais soluções para determinar o melhor modelo.

Na literatura existem vários trabalhos que utilizam o AFSA. Em [4] o AFSA é utilizado para otimizar redes *Radial Basis Funtion* (RBF) em aplicações de mercado de ações. No trabalho desenvolvido por Zhang [5] foi utilizado o AFSA e redes *MLP* treinadas com o algoritmo *backpropagation*. O AFSA tem sido utilizado com sucesso em várias outras aplicações como agrupamento em [6], ou problemas NP completos, como *Job Scheduling* em [7].

Neste trabalho utilizamos o algoritmo *Artificial Fish Swarm* (AFSA) incorporando características de Evolução Diferencial (DE) [8] para otimização de pesos e bias de *Extreme Learning Machines* (ELMs) [9]. Basicamente o AFSA é um algoritmo de busca aleatória que simula comportamentos dos peixes (Search, Swarm, Prey) e procura de forma efetiva a melhor solução global.

O algoritmo de evolução diferencial foi utilizado na otimização de ELMs em [10], onde a busca por boas soluções é realizada através de mutações dos indivíduos e transmissão de dados para um novo indivíduo através de operadores de cruzamento. Em [11]

é utilizado o algoritmo PSO para otimizar os pesos e bias do ELM, aplicado em problemas de regressão. A busca do PSO é realizada através das soluções encontradas por cada partícula e é direcionada pela melhor solução encontrada. O AFSA também utiliza a melhor solução encontrada por um peixe, porém utiliza informações sobre os peixes que estão dentro do seu campo visual.

O artigo será dividido da seguinte maneira: o algoritmo ELM é apresentado na seção 2.1, o AFSA será discutido em detalhes na seção 2.2 e analisaremos resumidamente o algoritmo de evolução diferencial na seção 2.3. O método proposto será apresentado na seção 3, o processo experimental será relatado na seção 4 juntamente com a discussão dos resultados e a conclusão será apresentada na seção 5.

2. Técnicas Utilizadas

2.1 Extreme Learning Machine

O ELM é um algoritmo para o treinamento de RNAs com apenas uma camada escondida. Apesar do seu treinamento rápido, é possível obter baixos erros de treinamento, assim como boa capacidade de generalização [9]. O ELM é baseado em operações matriciais, e para seu pleno desenvolvimento utiliza a matriz pseudo-inversa de Moore-Penrose [12]. O ELM pode ser resumido em três etapas básicas: primeiro, geram-se aleatoriamente os pesos de entrada e bias dos neurônios escondidos. Em seguida, calcula-se a matriz de saída da camada escondida. Finalmente, a matriz modificada é usada para obter a matriz de peso de saída, utilizando-se a matriz pseudo-inversa de Moore-Penrose.

Uma *RNA* com uma camada escondida e os Q neurônios escondidos podem ser matematicamente representados como:

$$\sum_{i=1}^{Q} \beta_i g(w_i x_j + b_i) = o_j$$

Onde, β é a matriz de pesos de saída que precisa ser determinada, $g(x)$ é a função de ativação utilizada, neste trabalho será utilizada a função sigmóide. Os pesos de entrada são representados por w_i, os bias são representados por b_i, e o_j é a matriz com as saídas esperadas para cada instância de treinamento x_j, onde $j = 1...N$ e $i = 1...Q$, podendo ser escrita da seguinte forma:

$$H\beta = T$$

, onde H é a matriz resultante da aplicação da função de ativação $g(x)$ do produto das entradas x_j pelos pesos intermediários w_i somados com o bias b_i. O propósito do algoritmo é determinar a matriz de pesos de saída β, e isso pode ser realizado aplicando a matriz pseudo-inversa de Moore-Penrose.

$$\hat{\beta} = H^{+}T$$

Partindo dos princípios de álgebra linear, sabemos que para calcular a inversa de uma matriz, um pré-requisito é que a matriz seja quadrada, ou seja, mesma dimensão de linhas e colunas. Como não podemos ter o controle sobre o número de dimensões que um dado problema pode ter, então precisamos aplicar um método mais robusto para o cálculo da matriz inversa.

O cálculo da matriz pseudo-inversa de Moore-Penrose, nada mais é do que o cálculo da inversa de uma matriz de dimensão qualquer. Baseando-se nos pesos e bias com valores atribuídos de forma aleatória, é possível calcular pesos de saída adequados à entrada apresentada.

O *ELM* é uma técnica que prima pela velocidade de treinamento e boa capacidade de generalização. Existem vários trabalhos recentes visando aperfeiçoar o *ELM* ou até propondo algoritmos diferenciados para o seu treinamento [13].

2.2 Artificial Fish Swarm Algorithm

O *Artificial Fish Swarm Algorithm* (AFSA), é um algoritmo de busca proposto em 2002, que tem inspiração no comportamento dos peixes [3]. Um peixe se locomove pela região visando alcançar um local onde haja maior concentração de comida (região ótima), e para isso leva em consideração características próprias mas também características dos outros peixes próximos.

Como parâmetros do algoritmo definimos o número de peixes P do cardume, a posição de cada peixe S_i que será definida como uma matriz de pesos e um vetor de bias, a capacidade de visualizar peixes próximos dos peixes $visual$, número de peixes vizinhos n_f, uma medida de distância D_{ij} entre peixes, a medida de nado de cada peixe $step$, um fator de lotação de uma determinada região δ e o número de tentativas de execução do comportamento *prey* $Trynumber$. Também definimos uma função objetivo $Y = f(x)$. A medida de distância utilizada pode ser a norma euclidiana L1 ou L2. Os peixes apresentam os seguintes comportamentos: Follow, Leap, Prey e Swarm, que serão descritos em detalhes a seguir.

2.2.1 Follow

Quando pequenos grupos de peixes encontram comida através da sua movimentação, os peixes vizinhos tendem a seguir-los ao local de melhor concentração de comida.

$$Follow(S_i) = S_i + rand() step \frac{S_{max} - S_i}{\|S_{max} - S_i\|}$$

caso a região não esteja lotada ($\frac{n_f}{N} < \delta$) e o o peixe melhor posicionado esteja em uma região de melhor concentração que o peixe atual ($Y_{max} > Y_i$). Caso contrário, a atualização ocorre da seguinte forma:

$$Follow(S_i) = Prey(S_i)$$

2.2.2 Prey

Instinto natural do peixe para encontrar comida, ou seja, o movimento dos peixes é direcionado pela concentração de comida na água. O peixe escolhe uma posição dentro do seu campo visual. Se depois de trynumber tentativas, o peixe não encontrar uma região com melhor concentração, ele se move aleatoriamente. Se o peixe selecionado, tiver concentração de comida maior, a atualização ocorre da seguinte forma:

$$Prey(S_i) = S_i + rand()step\frac{S_j - S_i}{\|S_j - S_i\|}$$

Caso o peixe não encontre nenhum vizinho com melhor concentração de comida após *trynumber* tentativas, ele se move aleatoriamente pelo espaço.

$$Prey(S_i) = Leap(S_i)$$

2.2.3 Swarm

Os peixes normalmente se movem em grupos para facilitar na caça e também para evitar eventuais situações de perigo. Então baseado na quantidade de peixes vizinhos, é calculado o centro S_c baseado nas informações de posição de cada peixe.

$$Swarm(S_i) = S_i + rand()step\frac{S_c - S_i}{\|S_c - S_i\|}$$

caso a região não esteja lotada ($\frac{n_f}{N} < \delta$) e o o peixe melhor posicionado esteja em uma região de melhor concentração que o peixe atual ($Y_c > Y_i$). Caso contrário, a atualização ocorre da seguinte forma :

$$Swarm(S_i) = Prey(X_i)$$

2.2.4 Leap

Movimentação do peixe aleatoria independente do cardume. Esta movimentação produz um comportamento estocástico dos peixes.

$$Leap(S_i) = S_i + rand()step$$

O comportamento que tiver a melhor avaliação será o comportamento utilizado pelo peixe para atualizar o seu estado.

2.3 Differential Evolution

O algoritmo de evolução diferencial proposto por Storn e Price [8] é conhecido como um dos mais eficientes algoritmos evolucionários. Esta técnica foi proposta para possibilitar que medidas não escalares possam ser facilmente utilizadas sem necessidade de conversão para o conjunto dos números binários. Ele funciona em três etapas fundamentais, Mutação, Cruzamento e a seleção. Dada uma população inicial $G_{i,t}$ a fase de mutação acontece da seguinte forma:

$$G_{i,t+1} = G_{r_1,t} + F(G_{r_2,t} - G_{r_3,t})$$

onde r_1,r_2 e r_3 são índices diferentes escolhidos aleatoriamente e F é um fator de amplitude para o termo $(G_{r_2,t} - G_{r_3,t})$. O fator de amplitude F possui faixa de valores no intervalo $[0..2]$.

Na fase de cruzamento um vetor V é criado com a mesma dimensão dos indivíduos. O vetor é inicializado da seguinte forma:

$$V_{ij} = \begin{cases} G_{ij,t} & \text{se } rand() < CR \quad \text{and } j = randIndex(i) \\ G_{ij,t+1} & \text{se } rand() > CR \quad \text{or } j \neq randIndex(i) \end{cases}$$

Onde CR é uma constante de cruzamento, inicializada pelo usuário e randIndex(i) é um índice selecionado aleatoriamente para garantir que pelo menos um elemento de $G_{i,t+1}$ seja atribuído a V_i. Os indivíduos com melhores avaliações serão selecionados para as gerações futuras, caso contrário os indivíduos antigos serão mantidos.

3. Algoritmo Proposto

O modelo híbrido proposto que chamamos de *Modified Artifical Fish Swarm for optimization of Extreme Learning Machines* (MAFSA-ELM) une características do AFSA e da DE, para otimizar os pesos do ELM. Antes de detalhar o funcionamento do modelo, é importante ressaltar alguns pontos.

Cada peixe será caracterizado por uma matriz de pesos intermediários W_{ij} e um vetor de bias b_k. A informação sobre os pesos de saída *a priori* é desconhecida e não será levada em consideração para o cálculo da distância entre peixes. A distância entre peixes será calculada de maneira similar a [3],

$$d_{pq} = \sum_{i=1}^{h}\sum_{j=1}^{n}(W_{ij}(p) - W_{ij}(q))^2 + \sum_{i=1}^{h}(b_{i0}(p) - b_{i0}(q))^2$$

onde h corresponde ao número de neurônios escondidos da rede e n ao número de atributos presentes em cada exemplo de treinamento. Cada peixe, é avaliado pelo seu desempenho no conjunto de validação gerado aleatoriamente em cada execução do algoritmo. A função de custo que iremos utilizar será o erro médio quadrático (*MSE*),

$$E = \frac{1}{2}\sum_{i=1}^{z}\sum_{k=1}^{u}(Y_k^i - T_k^i)^2$$

onde z corresponde ao número de exemplos dos dados de entrada e u o número de neurônios de saída que será igual ao número de classes do problema. A capacidade de generalização do algoritmo será avaliada utilizando o peixe que obtiver maior taxa de acerto no conjunto de validação em um conjunto de testes.

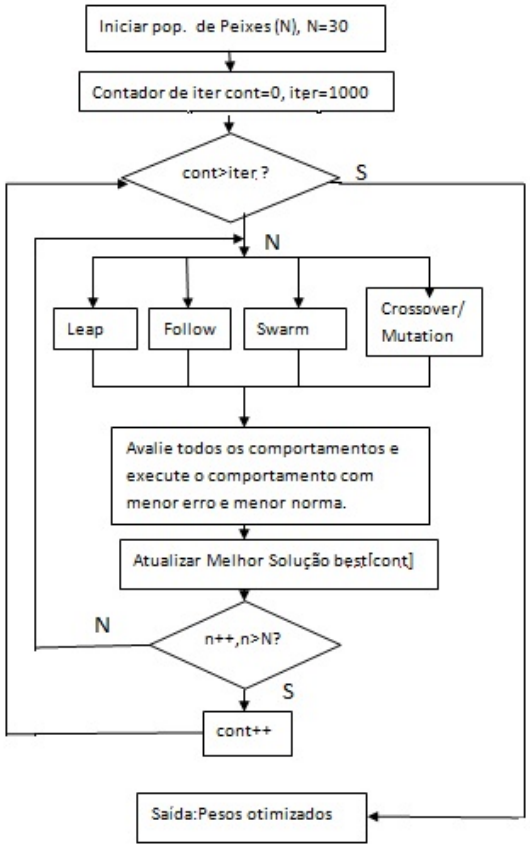

Figura 1: Fluxograma do funcionamento do algoritmo proposto

O algoritmo proposto realiza a busca levando em consideração aspectos da vizinhança de cada peixe através de uma variavel de campo de visão $visual$ e também informações sobre o melhor indivíduo dentro do campo visual de cada peixe. Além dos comportamentos presentes no AFSA ($Follow$,$Swarm$,$Leap$), adicionamos mais um comportamento que chamamos de *Crossover/Mutation* [8]. Esta modificação visa produzir mais variabilidade de soluções e potencializar a convergência mais rápida do algoritmo. Os operadores de cruzamento e mutação utilizam informações do cardume sem a restrição da variável *visual* que delimita os peixes vizinhos que podem ser encontrados. Portanto peixes que estiverem mais próximos de uma boa solução mas estiverem longe de um dado peixe, eles poderão ser considerados para esse comportamento. A execução do comportamento

ocorre de forma análoga ao que foi explicado na seção 2.3. As posições (pesos e bias) do cardume serão fornecidas para esta função, e através da seleção aleatória de indivíduos será realizada a etapa de mutação, e em seguida será realizada a etapa de cruzamento.

Um problema comum ao utilizar algoritmos de busca para otimizar pesos de ELMs é a falta de detecção de limites de busca. Segundo [11], os valores assumidos pelos pesos do ELM devem pertencer ao intevalo [-1,1], de forma que a busca continue com características aleatórias, porém com limites. Essa medida visa a reduzir a possibilidade de estagnação em mínimos locais. Para realizar a detecção de limites, utilizamos uma abordagem reflexiva, ou seja, os pesos que estiverem fora do intervalo serão invertidos.

$$W_{ij} = \begin{cases} \frac{1}{W_{ij}} & \text{se } |W_{ij}| > 1 \\ W_{ij} & \text{Caso contrário} \end{cases}$$

Conforme a figura 1, na primeira etapa do algoritmo, os parâmetros do AFSA são inicializados e o número de neurônios do ELM é escolhido. Cada peixe será caracterizado por uma matriz de pesos e um vetor de bias inicializados aleatoriamente. Em cada iteração do algoritmo, cada peixe escolherá o comportamento que produzir melhores resultados, obtidos pelo conjunto de validação. Se a solução obtida for melhor do que a solução atual, então a posição do peixe é atualizada. Porém segundo estudos realizados por Bartlett [14], apenas o erro de validação não garante a melhor capacidade de generalização da rede [10], e portanto adicionamos a norma dos pesos como critério de seleção.

Segundo [14],quanto menor forem os pesos de uma rede neural, melhor será sua capacidade de generalização, portanto se a solução for apenas um pouco inferior porém com menor norma, ela será utilizada. Após atualizar o estado de cada peixe, o melhor valor entre eles é guardado,e o contador de iterações incrementado. Ao final das iterações o melhor modelo é então apresentado, baseado no resultado obtido pela função de erro.

4 Experimentos

Foram realizados experimentos com 7 bases do UCI [15], onde os dados de cada base foram divididos em conjuntos de treinamento (50%), validação (25%) e teste(25%) gerados aleatoriamente durante 30 iterações.O conjunto de treinamento foi utilizado para a construção do modelo, e o conjunto de testes foi deixado separado para no final avaliar o modelo. Todos os atributos de todas as bases foram normalizados dentro do intervalo $[0..1]$. A especificação das bases pode ser encontrada na tabela 1.

Tabela 1: Descrição das bases utilizadas

Bases	Instâncias	Atributos	Classes
Diabetes	768	9	2
Ecoli	336	8	8
Ionosphere	351	35	2
Sonar	208	61	2
Glass	214	10	6
Vehicle	846	19	4
Horse	368	28	3

Para avaliarmos o desempenho do algoritmo proposto, utilizamos o algoritmo ELM traidicional, redes RBF [1], o E-ELM proposto em [10], e o *Evolutionary Extreme Learning Machine Based on Paticle Swarm Optimization*(PSO-ELM) [11], nas bases Diabetes, Ecoli, Glass, Ionosphere, Sonar e Vehicle. No final de cada simulação foi aplicado o teste de hipótese de Wilcoxon com 95% de significância para detectar diferenças significativas entre seus resultados.

Em outros trabalhos, verificamos diversos tipos de configurações [6], [7], [16], [17] para os parâmetros do AFSA, e verificamos que o parâmetro $visual$ é dependente do problema a ser tratado. Neste trabalho o utilizamos o valor de $visual$ igual a média de todas as distâncias iniciais entre os peixes, e resultados promissores foram produzidos com essa estratégia de configuração. Além do $visual$, inicializamos também o número de peixes $N = 30$, fator de lotação $\delta = 0.8$, o fator de correção $step = 0.6$, número de tentativas $tryNumber = 5$, iterações $iter = 50$, fator de amplitude para a mutação $F = 1$ e $CR = 0.5$ (valor usado normalmente conforme [10]).

Para os experimentos os parâmetros do PSO-ELM utilizaremos a configuração descrita em [11] com algumas modificações. Os valores das constantes $C1$ e $C2$ foram mantidos com $C1 = 2$ e $C2 = 2$, utilizamos o fator de inércia $w = 0.9$,modificamos o número de partículas para $N = 30$ executando o algoritmo por 50 iterações. Os parâmetros do E-ELM são utilizados conforme descrito em [10], $F = 1$, $CR = 0.5$, o número de indivíduos foi aumentado de 20 para 30 neste trabalho.

Os resultados obtidos pelo processo experimental são mostrados na tabela 2. Os resultados apresentados correspondem as médias e desvios padrão (DP) das taxas de acerto no conjunto de testes obtidas em cada execução dos algoritmos, juntamente com os neurônios da camada escondida da rede Q. As simulações foram realizadas com 10,15 e 20 neurônios escondidos, e o melhor resultado foi selecionado.

Tabela 2: Resultado das simulações para os algoritmos ELM, RBF Network, E-ELM, PSO-ELM e MAFSA-ELM

Métodos	ELM		RBF Network		E-ELM		PSO-ELM		MAFSA-ELM	
Bases	Média $\pm DP$	Q	Média$\pm DP$	Q	Média$\pm DP$	Q	Média$\pm DP$	Q	Média$\pm DP$	Q
Diabetes	76.98 ± 1.48	10	76.63 ± 2.06	10	76.73 ± 3.51	10	75.32 ± 2.79	10	**77.22 ± 2.80**	10
Ecoli	82.84 ± 3.04	15	82.71 ± 4.65	15	83.65 ± 4.20	10	82.50 ± 4.97	20	**84.24 ± 3.76**	15
Ionosphere	84.81 ± 3.04	20	85.7 ± 4.47	20	86.36 ± 4.12	15	85.22 ± 4.87	15	**88.29 ± 4.22**	20
Sonar	70.62 ± 4.33	20	69.58 ± 6.19	15	73.26 ± 6.68	20	73.39 ± 5.48	20	**73.71 ± 5.15**	20
Glass	62.33 ± 4.45	20	61.66 ± 5.17	20	63.14 ± 5.07	20	63.27 ± 6.53	15	**64.27 ± 5.92**	20
Vehicle	73.27 ± 2.68	20	74.66 ± 2.16	20	74.73 ± 2.88	20	**75.73 ± 1.95**	20	74.91 ± 2.25	20
Horse	65.06 ± 2.93	20	65.11 ± 3.46	20	67.57 ± 3.43	20	66.46 ± 4.37	20	**68.07 ± 4.59**	20

Realizando uma análise empírica sobre os dados da tabela 2, verificamos que o algoritmo proposto (MAFSA-ELM) obteve resultados superiores em seis (Diabetes,Ecoli,Ionosphere,Glass,Sonar,Horse) das sete bases testadas. Aplicamos o teste de Wilcoxon com 95% de confiança nos resultados obtidos nas 30 execuções em todas as bases, para obtermos uma análise mais realista do seu desempenho.

Baseado nos resultados do teste de Wilcoxon, o ELM e o RBF obtiveram desempenhos similares em todas as bases. Na base diabetes, o MAFSA-ELM obteve resultados similares aos demais algoritmos testados, sendo superior ao PSO-ELM apenas. Na base ecoli observamos que o algoritmo proposto obteve resultados superiores à todos os algoritmos com exceção do E-ELM. Na base Ionosphere o MAFSA-ELM necessitou de mais neurônios escondidos, porém obteve resultados superiores à todos os algoritmos, que não obtiveram melhoras no desempenho ao aumentar o número de neurônios. Nas bases Sonar, Glass e Vehicle, o MAFSA-ELM obteve resultados similares ao E-ELM e ao PSO-ELM. Na base Horse, o desempenho do MAFSA-ELM foi similiar ao E-ELM, sendo superior aos demais algoritmos.

De forma geral,o algoritmo proposto obteve resultados competitivos, superiores em alguns casos, em relação às técnicas de otmização de ELMs utilizadas para comparação.

5. CONCLUSÃO

Neste trabalho foi proposto um novo algoritmo híbrido para otimização de pesos de RNAs com uma única camada escondida treinadas com ELM. O Extreme Learning Machine é uma técnica de treinamento rápido e utiliza a regra dos quadrados mínimos (Matriz pseudo-inversa de Moore-Penrose) para calcular os pesos de saída da rede. Inicializar os pesos aleatoriamente no extreme learning machine pode levar a rede a bons resultados, porém com a devida otimização, pode-se conseguir resultados melhores.

O algoritmo proposto foi o MAFSA-ELM *Modified Artifical Fish Swarm for optimization of Extreme Learning Machines*, que é baseado no comportamento dos peixes realizando buscas com características aleatórias, e foi adicionado um comportamento baseado em evolução diferencial, para melhorar a qualidade das soluções e para convergência mais rápida dos modelos (Extreme learning machines). Também foi incorporado ao modelo uma abordagem para detecção de limites de busca nos pesos do ELM.

O MAFSA-ELM foi testado em sete bases do UCI (Diabetes, Sonar, Ecoli, Ionosphere, Vehicle, Horse e Glass) e obteve resultados superiores em alguns casos sob o teste de Wilcoxon, e utilizando abordagens empíricas, foi superior em seis das sete bases.

Como trabalhos futuros, realizaremos um estudo sobre a composição de ensembles utilizando o MAFSA-ELM, ressaltando pontos como desempenho e diversidade dos modelos gerados e realizaremos testes comparativos com trabalhos já publicados [18].

REFERÊNCIAS

[1] S. Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR Upper Saddle River, NJ, USA, 1994.

[2] A. Engelbrecht. *Fundamentals of computational swarm intelligence*, volume 1. Wiley NY, 2005.

[3] C. Wang, C. Zhou and J. Ma. "An improved artificial fish-swarm algorithm and its application in feed-forward neural networks". In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 5, pp. 2890–2894. IEEE, 2005.

[4] D. Niu, W. Shen and Y. Sun. "RBF and Artificial Fish Swarm Algorithm for short-term forecast of stock indices". In *Communication Systems, Networks and Applications (ICCSNA), 2010 Second International Conference on*, volume 1, pp. 139–142. IEEE, 2010.

[5] M. Zhang, C. Shao, F. Li, Y. Gan and J. Sun. "Evolving neural network classifiers and feature subset using artificial fish swarm". In *Mechatronics and Automation, Proceedings of the 2006 IEEE International Conference on*, pp. 1598–1602. IEEE, 2006.

[6] S. He, N. Belacel, H. Hamam and Y. Bouslimani. "Fuzzy Clustering with Improved Artificial Fish Swarm Algorithm". In *Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on*, volume 2, pp. 317–321. IEEE, 2009.

[7] K. Zhu and M. Jiang. "The optimization of job shop scheduling problem based on Artificial Fish Swarm Algorithm with tabu search strategy". In *Advanced Computational Intelligence (IWACI), 2010 Third International Workshop on*, pp. 323–327. IEEE.

[8] R. Storn and K. Price. "Differential evolution–a simple , efficient heuristic for global optimization over continuous spaces". *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[9] G. Huang, Q. Zhu and C. Siew. "Extreme learning machine: theory and applications". *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[10] Q. Zhu, A. Qin, P. Suganthan and G. Huang. "Evolutionary extreme learning machine". *Pattern recognition*, vol. 38, no. 10, pp. 1759–1763, 2005.

[11] Y. Xu and Y. Shu. "Evolutionary Extreme Learning Machine–Based on Particle Swarm Optimization". *Advances in Neural Networks-ISNN 2006*, pp. 644–652, 2006.

[12] C. Rao and S. Mitra. *Generalized inverse of matrices and its applications*. Wiley NY, 1971.

[13] E. Soria-Olivas, J. Gomez-Sanchis, J. Martin, J. Vila-Frances, M. Martinez, J. Magdalena and A. Serrano. "BELM: Bayesian extreme learning machine". *Neural Networks, IEEE Transactions on*, vol. 22, no. 3, pp. 505–509, 2011.

[14] P. Bartlett. "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network". *Information Theory, IEEE Transactions on*, vol. 44, no. 2, pp. 525–536, 1998.

[15] C. Blake and C. Merz. "UCI repository of machine learning databases". 1998.

[16] H. Ma and Y. Wang. "An artificial fish swarm algorithm based on chaos search". In *2009 Fifth International Conference on Natural Computation*, pp. 118–121. IEEE, 2009.

[17] M. Jiang and Y. Cheng. "Simulated annealing artificial fish swarm algorithm". In *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, pp. 1590–1593. IEEE, 2010.

[18] H. Escalante, M. Montes and E. Sucar. "Ensemble particle swarm model selection". In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pp. 1–8. IEEE.