10th Brazilian Congress on Computational Intelligence (CBIC'2011), November 8 to 11, 2011, Fortaleza, Ceará Brazil

OTIMIZAÇÃO DE REDES NEURAIS USANDO GSO COOPERATIVOS COM DECAIMENTO DE PESOS

D. N. G. Silva e T. B. Ludermir

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Av. Jornalista Anibal Fernandes, s/n, 50.740-560 – Recife – PE – Brasil
{dngs, tbl}@cin.ufpe.br

Resumo – Treinamento de Redes Neurais Artificiais (RNAs) é uma tarefa complexa de grande importância em problemas de aprendizado supervisionado. Algoritmos Evolutivos (AEs) são amplamente utilizados como técnicas de otimização global e estas abordagens têm sido usadas para as RNAs executarem várias tarefas. Recentemente, um algoritmo de otimização, chamado de *Group Search Optimizer* (GSO), foi proposto e inspirado pelo comportamento de busca dos animais. Neste artigo apresentamos duas novas abordagens híbridas: GSO-S_k-WD e GSO-H_k-WD. Os GSO cooperativos são baseados no paradigma de dividir e conquistar, empregando o comportamento cooperativo entre os grupos GSOs para melhorar o desempenho do GSO padrão. Também aplicamos a estratégia de decaimento de pesos (WD - *Weight Decay*) para aumentar o poder de generalização das redes. Os resultados mostram que os GSO cooperativos são capazes de conseguir melhor desempenho do que o GSO tradicional para problemas de classificação dos conjuntos de dados de Câncer, Diabetes, Ecoli e Vidros.

Palavras Chave – Redes Neurais Artificiais, Sistemas Híbridos, *Group Search Optimizer*, Treinamento de Redes Neurais, Limites do Espaço de Busca.

1 Introducão

Redes Neurais Artificiais (RNAs) são conhecidas como aproximadores universais e modelos computacionais com características particulares, como a adaptabilidade, a capacidade de aprender por exemplos e a capacidade de organizar ou generalizar os dados.

O processo de treinamento da *Multilayer Perceptron* (MLPs) para problemas de classificação de padrões é composto por duas tarefas difíceis, a primeira é a seleção de uma arquitetura apropriada para o problema, e a segunda é o ajuste dos pesos das conexões da rede [14]. Algoritmos tradicionais de aprendizagem, tais como a técnica baseada no gradiente, chamada *Backpropagation* (BP) e sua variante Levenberg-Marquardt (LM) têm sido amplamente utilizados na formação das MLPs, mas estas abordagens podem ficar presas em mínimos locais. Atualmente, pode-se dizer que a área da Inteligência Artificial está em um estágio onde o objetivo final é a integração de diferentes técnicas para construir sistemas mais robustos, os chamados Sistemas Inteligentes Híbridos (SIHs). A motivação para esses sistemas serem desenvolvidos dá-se ao fato de que eles possibilitam a resolução de problemas complexos que só poderiam ser resolvidos quando tratados por cada uma das suas sub-tarefas, assim como unir as vantagens e superar as limitações individuais de diferentes técnicas. Técnicas de pesquisa global, tais como Algoritmos Evolutivos, Algoritmos Genéticos, Busca Tabu, Otimização por Colônia de Formigas e Otimização por Enxame de Partículas (PSO - *Particle Swarm Optimization*), com a possibilidade de alargar o espaço de busca, na tentativa de evitar mínimos locais, tem sido amplamente combinados com Redes Neurais para realizar várias tarefas como: inicialização dos pesos das conexões, conexões dos pesos de treinamento, otimização da arquitetura, entre outros parâmetros da Rede Neural. Recentemente, uma nova técnica de otimização inspirada na natureza foi proposta: *Group Search Optimizer* (GSO) [2].

O algoritmo GSO é um novo algoritmo de Inteligência de Enxame [8] para problemas de otimização. O algoritmo é baseado em um modelo de forrageiro social, Produtor-Dependente (PS - *Produtor-Scrounger*) modelo descrito em [1], o que pressupõe que os membros do grupo de pesquisa ou "encontram" (*producer*) ou "juntam" (*scrounging*) oportunidades, diferentemente das metáforas usadas pelo ACO e PSO [7, 9]. Algumas aplicações práticas têm sido feitas usando o algoritmo GSO. Em [4], a estratégia de *Group Search Optimizer* para treinamento de uma RNA foi utilizada para diagnóstico de câncer de mama. Em [3] foi apresentado um método de otimização multi-objetivo, onde o GSO com vários produtores (GSOMP) foi aplicado para o posicionamento ideal do sistema *Flexible AC Transmission System* (FACTS) para minimizar a perda real de energia e melhorar seu perfil de tensão. Em [5] foi aplicada uma RNA treinada com GSO para monitoramento da condição das máquinas de ultra-som.

Estratégias cooperativas que têm obtido bons resultados em problemas de otimização numérica são baseadas na divisão do espaço de busca. Assim, o espaço de busca original de dimensão n é dividido em $1 \leq n \leq K$ partições de dimensão d, com $d = K \times n$ [10, 12]. Em [18] foi introduzido esta idéia por meio de um algoritmo genético co-evolucionário (CCGA, do inglês *coevolutionary genetic algorithm*) para função de otimização, que consiste em uma decomposição de um problema n-dimensional (*n* variáveis), em problemas *n* 1-dimensional, cada um abordado por uma diferente sub-população (espécie). [19] apresentou um algoritmo para a integração de modelos substitutos da função de *fitness* com os procedimentos de buscas co-evolutivas. Em [10, 11], o modelo de Potter [18] foi estendido, onde duas abordagens cooperativas foram apresentadas: CPSO-S_K e CPSO-H_K. Neste artigo, apresentamos duas novas abordagens híbridas do GSO, inspiradas nos estudos feitos pelos artigos citados neste parágrafo, onde faz uso da cooperação entre os diferentes grupos do GSO. Em nosso trabalho adotamos 5 partições, combinadas com a heurística de decaimento de pesos [13] para melhorar o processo de otimização dos pesos das

MLPs: O *Group Search Optimizer* Cooperativo (CGSO-S_K-WD) e o *Group Search Optimizer* Híbrido Cooperativo (CGSO-H_K-WD). O comportamento cooperativo é obtido pela estratégia de dividir e conquistar, onde cada grupo é responsável por um conjunto limitado de variáveis do problema, e a solução final é encontrada através da combinação de soluções encontradas por cada grupo do GSO. Estas técnicas são extensões dos trabalhos apresentados em [18] e [10, 11] para o contexto do algoritmo GSO. Os experimentos foram feitos com os algoritmos GSO e Levenberg-Marquardt tradicionais e nossas abordagens para problemas reais de classificação obtidos a partir do repositório UCI [22].

Este artigo está organizado em: Seção II apresenta o padrão de pesquisa do algoritmo GSO [2], a heurística de decaimento de pesos [13, 16] e as estratégias cooperativas. Em seguida, nossas abordagens de busca cooperativa usando o GSO são apresentadas na seção III e os resultados experimentais são mostrados na seção IV. As conclusões são dadas na Seção V.

2 Preliminares

Nesta seção serão apresentados os algoritmos e técnicas que formaram os modelos propostos neste artigo. Estas serão apresentadas na sua forma original e assim possamos entender a composição do modelo híbrido proposto.

2.1. Group Search Optimizer (GSO)

O GSO [2] é um algoritmo de enxame inteligente inspirado pelo comportamento social de pesquisa dos animais e na teoria de vida em grupo. O GSO emprega o modelo *producer-scrounger* (PS) [1]. Neste modelo, presume-se que há duas estratégias de forrageio em grupos: (1) produzindo, por exemplo, em busca de alimento, e (2) juntar (arrecadar), por exemplo, unindo recursos descobertos por outros. A população com tamanho N do GSO é chamada de *grupo* e cada indivíduo da população é chamado de *membro*. No espaço de busca n dimensional, o membro *ith* da iteração *kth* tem uma posição corrente $\mathbf{X}^k_i \in \Re^n$ e um ângulo de cabeça $\boldsymbol{\varphi}^k_i = (\boldsymbol{\varphi}^k_{i\,1},..., \boldsymbol{\varphi}^k_{i\,(n-1)}) \in \Re^{n-1}$. Este ângulo de cabeça limita o até onde o indivíduo pode girar seu ângulo de busca. Cada direção de busca do membro *ith*, que é um vetor unitário $\mathbf{D}^k_i(\boldsymbol{\varphi}^k_i) = (d^k_{i1},..., d^k_{in}) \in \Re^n$ pode ser calculada a partir de $\boldsymbol{\varphi}^k_i$ como segue:

$$\begin{aligned} d^k_{i1} &= \prod_{q=1}^{n-1} \cos(\varphi^k_{iq}) \\ d^k_{ij} &= \sin(\varphi^k_{i(j-1)}) \cdot \prod_{q=j}^{n-1} \cos(\varphi^k_{iq}) \quad (j = 2,..., n-1) \\ d^k_{in} &= \sin(\varphi^k_{i(n-1)}). \end{aligned} \tag{1}$$

No GSO, um grupo é composto por três tipos de membros: os produtores, os dependentes e os membros dispersos [2]. Existe apenas um produtor em cada busca do GSO e os demais membros são dependentes ou membros dispersos. Todos os dependentes irão juntar o recurso encontrado pelo produtor. Durante cada iteração do GSO, um membro do grupo que encontrou o melhor valor de *fitness* é escolhido como produtor. Na iteração *kth*, o produtor $\mathbf{X}p$ se comporta como se segue:

Passo 1 - O produtor irá varrer a zero grau (ou seja, ele varrerá os espaço a sua volta sem mudar sua ângulo) e lateralmente por dois pontos no campo de digitalização da amostragem, um ponto a zero grau (2), um ponto no lado direito (3) e um ponto no lado esquerdo (4):

$$\mathbf{X}_z = \mathbf{X}^k_p + r_1 l_{max} \mathbf{D}^k_p(\boldsymbol{\varphi}^k) \tag{2}$$

$$\mathbf{X}_r = \mathbf{X}^k_p + r_1 l_{max} \mathbf{D}^k_p(\boldsymbol{\varphi}^k + \mathbf{r}_2 \theta_{max}/2) \tag{3}$$

$$\mathbf{X}_l = \mathbf{X}^k_p + r_1 l_{max} \mathbf{D}^k_p(\boldsymbol{\varphi}^k - \mathbf{r}_2 \theta_{max}/2) \tag{4}$$

onde $r_1 \in \Re^1$ é um número aleatório normalmente distribuído com média 0 e desvio padrão 1, $r_2 \in \Re^{n-1}$ é uma seqüência aleatória uniformemente distribuída no intervalo (0, 1), $\theta_{\max} \in \Re^1$ é o ângulo de busca máxima, e $l_{max} \in \Re^1$ é a distância máxima de busca.

Passo 2 - O produtor vai encontrar o melhor ponto (melhor valor de *fitness*) entre todos os pontos avaliados na etapa 1. Se o melhor ponto tem um valor melhor do que a sua posição atual, então ele vai voar para este melhor ponto encontrado. Caso contrário ele vai ficar na sua posição atual e virar a cabeça para um novo ângulo que será gerado aleatoriamente para tentar uma nova busca:

$$\boldsymbol{\varphi}^{k+1}_i = \boldsymbol{\varphi}^k_i + \mathbf{r}_2 \alpha_{max} \tag{5}$$

onde $\alpha_{\max} \in \Re^1$ é o ângulo máximo de viragem.

Passo 3 - Se o produtor não pode achar uma melhor área após *a* iterações, ele vai virar a cabeça e voltar para zero grau:

$$\varphi^{k+a} = \varphi^{k} \qquad (6)$$

onde $a \in \Re^1$ é uma constante dada por *round*($\sqrt{n+1}$) [2]. Em cada iteração, um número de membros do grupo é selecionado como dependentes, que continuam procurando oportunidades para juntar os recursos encontrados pelo produtor, adotando a estratégia de cópia de área [1]. Na iteração *k*th, o comportamento de copiar área do dependente *ith*, pode ser modelado como:

$$\mathbf{X}_i^{k+1} = \mathbf{X}_i^{k} + \mathbf{r}_3 \circ (\mathbf{X}_p^{k} - \mathbf{X}_i^{k}) \qquad (7)$$

onde $r_3 \in \Re^n$ é uma seqüência aleatória uniforme em (0, 1).

Os demais membros do grupo serão dispersos a partir de suas posições atuais. Caminhadas aleatórias são adotadas por esses membros dispersos como a estratégia adotada por [6]. Na iteração *kth* se o membro do grupo *ith* é selecionado como um membro disperso, ele gera um ângulo de cabeça aleatória $\boldsymbol{\varphi}_i$ usando (5) e, em seguida, ele escolhe uma distância aleatória:

$$l_i = a \,.\, r_1 l_{max} \qquad (8)$$

e move para um novo ponto:

$$\mathbf{X}_i^{k+1} = \mathbf{X}_i^{k} + l_i \mathbf{D}_i^{k} (\boldsymbol{\varphi}^{k+1}) \qquad (9)$$

No GSO, quando um membro está fora do espaço de busca, ele volta para sua posição anterior, dentro do espaço de busca, restringindo a pesquisa a um trecho rentável. O algoritmo padrão do GSO está presente em [2].

2.2.Decaimento de Pesos (WD)

Decaimento de pesos foi inicialmente sugerido como uma heurística para melhorar o algoritmo de retropropagação das bias de uma robusta rede neural que é insensível a ruídos [13]. Esta aplicação usou um esquema adaptativo para a determinação do coeficiente de regularização ($\lambda(t)$), como descrito em [16]. Cada membro tem seu próprio coeficiente $\lambda_i(t)$ ajustado de forma adaptativa de acordo com seu erro médio. Após determinar a posição do indivíduo através da GSO padrão, o termo é aplicado à deterioração da posição atual do membro, de acordo com a equação 10.

$$\mathbf{X}_i(t+1) = \mathbf{X}_i(t+1) - \lambda_i(t+1)\,\mathbf{X}_i(t+1) \qquad (10)$$

A nova função de custo é dada por:

$$C = E_i(t) + \frac{\lambda_i(t)}{2} \|\mathbf{X}_i\|^2 \qquad (11)$$

Onde $E_i(t)$ denota o erro da rede neural no tempo *t* relacionado ao indivíduo *i* e $\|\mathbf{X}_i\|^2$ é a norma ao quadrado de $\mathbf{X}_i$. A cada iteração, cada $\lambda_i(t)$ é atualizado da seguinte forma:

$$\lambda_i(t+1) = \begin{cases} \lambda_i(t) + INC, & if \quad E_i(t) < \overline{E}_i(t) \\ \lambda_i(t) - INC, & if \quad E_i(t) \geq \overline{E}_i(t) \\ \lambda_i(t), & senão \end{cases} \qquad (12)$$

Onde *INC* é o valor do incremento do coeficiente de regularização e $\overline{E}_i(t)$ denota o erro médio obtido pelo indivíduo *ith* até um tempo *t*. O mecanismo de decaimento de pesos atua em uma arquitetura de rede diferencialmente em direção a zero, reforçando grandes pesos e enfraquecendo conexões de pequenos pesos.

2.3.Abordagens Cooperativas

A cooperação envolve um conjunto de indivíduos que interagem através da comunicação de informação entre si, enquanto resolvem problemas. Uma das primeiras formas de cooperação foi estudada com base na evolução das ilhas [20], onde cada ilha corresponde a uma sub-população isolada, pesquisando em uma área separada do espaço de solução. Depois de *m* iterações, as ilhas enviam e recebem $1 \leq z \leq 5$ indivíduos que promovem um intercâmbio de informações entre as ilhas. A forma de cooperação é baseada na abordagem celular [20], onde o conceito de vizinhança é usado.

Uma estratégia de cooperação que tem obtido melhores resultados em problemas de otimização numérica é a baseada na divisão do espaço de busca. Esta estratégia foi inicialmente introduzida por [18], por meio de um algoritmo genético co-evolucionário cooperativo (CCGA). A função de *fitness* para um membro da população é obtida pela combinação de suas

variáveis com as melhores soluções encontradas até o momento pelas sub-populações restantes. No PSO esta abordagem foi introduzida pelo algoritmo CPSO-S_k [10, 11].

3. Group Search Optimizer Cooperativos

Esta seção apresenta as nossas duas novas abordagens híbridas do GSO baseadas na cooperação: CGSO-S_k-WD e CGSO-H_k-WD. Aplicamos também a heurística de decaimento de pesos para aumentar o poder de generalização das nossas metodologias. Em nossa abordagem, adotamos o tratamento de absorção para os membros que escapam do espaço de busca. Em CGSO-S_K-WD o espaço de dimensão *n* da busca tem sido dividido em partições *K* de dimensão *d* em que uma sub-pesquisa é executada. Formalmente, a cooperação entre as sub-populações é feita através da concatenação dos sub-indivíduos atuais que desejam avaliar e os melhores sub-indivíduos até o momento, das demais partições, nas posições correspondentes. Esta composição é representada pelo contexto do vetor resultante da função $\boldsymbol{b}(j, vec)$ expressada na equação (13), com *vec* como o sub-individuo da sub-população *j* que queremos avaliar e $G_i.\mathbf{X}_p$ como o melhor sub-individuo da sub-população *i*.

$$\boldsymbol{b}(j, vec) \equiv (G_1.\mathbf{X}_p, \ldots, G_{j-1}.\mathbf{X}_p, \ldots, G_{j+1}.\mathbf{X}_p, \ldots, G_K.\mathbf{X}_p) \qquad (13)$$

Assim, podemos descrever o algoritmo CGSO-S_k como sendo realizado pela cooperação de GSOs que otimiza cada uma das partições *K* do espaço de busca. Nosso método CGSO-H_k é baseado no algoritmo CPSO-H_k apresentado em [10, 11]. A troca de informações é realizada de tal forma que alguns membros de uma metade da população do algoritmo são substituídos pela melhor solução descoberta até o momento pela outra metade do algoritmo. Depois de uma iteração da metade do algoritmo CGSO-S_k (o grupo G_j), o vetor do contexto $\boldsymbol{b}(1, G_1.\mathbf{X}_p)$ é usado para substituir uma partícula escolhida aleatoriamente da metade do algoritmo GSO (o grupo *Q*). Isso é seguido por uma iteração do componente do grupo *Q*, que produz um novo global, $Q.\mathbf{X}_p$. Esse vetor é então dividido em sub-vetores de dimensão apropriada e utilizado para substituir as posições das partículas escolhidas aleatoriamente nos grupos G_j. O algoritmo não substitui o $\mathbf{X}_p$ de nenhum grupo [11]. Tabela 1 e Tabela 2 mostram os pseudocódigos do CGSO-S_k-WD e do CGSO-S_k-WD, respectivamente. Nessas abordagens, cada membro *i* do grupo é composto por um conjunto de pesos entre a camada de entrada e nós ocultos ($\mathbf{W}_1$), bias ocultas ($\boldsymbol{\Theta}_1$), um conjunto de pesos entre a camada escondida e a camada de saída ($\mathbf{W}_2$), e bias de saída ($\boldsymbol{\Theta}_2$) [4]:

$$\mathbf{X}_i = [\mathbf{W}^i_1, \boldsymbol{\Theta}^i_1, \mathbf{W}^i_2, \boldsymbol{\Theta}^i_2]$$

Para cada membro, todas as variáveis são inicializadas aleatoriamente dentro do intervalo [-1, 1]. A função de fitness adotada é o erro médio quadrático (MSE, acrónimo de *mean squared error*) no conjunto de validação:

$$E = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{C}\left(t^n_k - o^n_k\right)^2 \qquad (14)$$

Onde t^n_k é a saída desejada relacionada ao indivíduo *ith* referente à classe *kth*, o^n_k é a saída obtida pela RNA para o indivíduo *ith* referente à classe *kth* e *C* é o número de nós de saída.

1. $k = 0$;
2. **PARA** (cada grupo $j \in K$)
3. Inicialize aleatoriamente posições $G_j.\mathbf{X}_i$ ângulo de cabeça $\boldsymbol{\varphi}_i$ de todos os membros;
4. Calcule o valor inicial da função de custo dos membros: $f(G_j.\mathbf{X}_i)$;
5. **FIM PARA**
6. **ENQUANTO** (Se as condições finais não forem atendidas)
7. **PARA** (cada grupo $j \in K$)
8. **PARA** (cada membro *i* do grupo)
9. **Escolher o produtor**: encontre o produtor $\mathbf{X}_p$ do grupo;
10. **Performance do produtor:** Comportamento do produtor como nos passos 1, 2 e 3;
11. **Performance do dependente**: Selecione uma porcentagem dos restantes dos membros para ser dependentes;
12. **Performance dos membros dispersos**: Os restantes dos membros irão executar a performance dos membros dispersos:
    a. Gerar o ângulo de cabeça aleatório usando (5);
    b. Escolha a distância l_i usando (8) e mova para o novo ponto usando (9);
13. **Avaliação do *Fitness***: Calcule o valor do *fitness* do membro atual: $f(\mathbf{X}_i)$;
14. **Atualização do coeficiente de regularização**: Atualizar todos os decaimentos de pesos usando (12);
15. **FIM PARA**
16. **FIM PARA**
17. $k = k + 1$;
18. **FIM ENQUANTO**

Tabela 1 – Pseudocódigo do algoritmo CGSO-S_k-WD

Sobre o algoritmo 1 e 2, podemos destacar que o valor de *fitness* é calculado através do erro da RNA, e o critério de parada adotado foi este erro chegar ao valor mínimo igual a 0, além do número máximo de iteração do GSO e da RNA.

1. $k = 0$;
2. **PARA** (cada grupo $j \in K$)
3. Inicialize aleatoriamente posições $G_j.\mathbf{X}_i$ e ângulos de cabeça $\boldsymbol{\varphi}_i$ de todos os membros;
4. Calcule o valor da função de custo iniciais dos membros: $f(G_j.\mathbf{X}_i)$;
5. **FIM PARA**
6. Inicialize o *n*-dimensional GSO: Q
7. **ENQUANTO** (as condições finais não forem atendidas)
8. **PARA** (cada grupo $j \in K$)
9. **PARA** (cada membro i do grupo)
10. **Escolher produtor**: Encontre o produtor $\mathbf{X}_p$ do grupo;
11. **Performance do produtor:** Comportamento do produtor como nos passos 1, 2 e 3;
12. **Performance do dependente**: Selecione uma porcentagem dos membros para executar a performance dos dependentes;
13. **Performance de dispersão**: Os restantes dos membros irão dispersar:
    a. Gerar um ângulo de cabeça aleatório usando (5);
    b. Escolher a distância l_i usando (8) e mova para um novo ponto usando (9);
14. **Evolução do Fitness**: Calcule o calor da função de custo do membro atual: $f(\mathbf{X}_i)$;
15. **Atualizar coeficiente regularização**: atualizar decaimentos de pesos usando (12);
16. **FIM PARA**
17. **FIM PARA**
18. Selecione aleatoriamente $i \sim U(1, N/2 \mid Q.\mathbf{X}_i \neq Q.\mathbf{X}_p$
19. $Q.\mathbf{X}_i = \boldsymbol{b}(1, G_1.\mathbf{X}_p)$
20. **PARA** (cada membro i do grupo)
21. **Escolher produtor**: Encontre o produtor $\mathbf{X}_p$ do grupo;
22. **Performance do produtor:** comportamento do produtor como nos passos 1, 2 e 3;
23. **Performance dos dependentes**: Selecione uma porcentagem dos membros para executar a performance dos dependentes;
24. **Performance de dispersão**: Os restantes dos membros irão dispersar:
    b. Gerar um ângulo de cabeça aleatório usando (5);
    c. Escolher a distância l_i usando (8) e mova para um novo ponto usando (9);
25. **Evolução do Fitness**: Calcule o valor da função de custo do membro atual: $f(\mathbf{X}_i)$;
26. **Atualizar coeficiente de regularização**: Atualizar os decaimentos pesos usando (12);
27. **FIM PARA**
28. **PARA** cada grupo $j \in [1\ldots K]$:
29. Selecione aleatoriamente $i \sim U(1, s/2 \mid G_j.\mathbf{X}_i \neq G_j.\mathbf{X}_p$
30. $G_j.\mathbf{X}_i = Q.\mathbf{X}_p$
31. **FIM PARA**
32. $k = k + 1$;
33. **FIM ENQUANTO**

Tabela 2 – Pseudocódigo do algoritmo CGSO-H_k-WD

4. Resultados e Discussões

Nesta seção, nós comparamos o tempo de treinamento e a *accuracy* de teste do GSO tradicional com o LM e nosso novo CGSO. Todos os programas são executados no MATLAB 6.0, e o algoritmo LM é fornecido pelo toolbox de redes neurais do MATLAB. Um conjunto de validação é usado em todas as metodologias avaliadas para evitar *overfitting*. Para avaliar todos estes algoritmos foram utilizados quatro conjuntos de dados de classificação: câncer, diabetes, vidros e ecoli, obtido a partir do repositório UCI [22]. Estes conjuntos de dados apresentam diferentes graus de dificuldades, números de classes diferentes e números de exemplos e características diferentes. Para estas abordagens, consideraram-se redes com apenas uma camada escondida, com seis neurônios nesta camada, realizando o ajuste dos pesos correspondentes. Estes parâmetros forma escolhidos após alguns experimentos com outros valores para estes parâmetros, sendo assim os valores escolhidos foram escolhidos de acordo com os melhores resultados obtidos nos experimentos.

As métricas de avaliação utilizadas são de uma análise empírica e análise de variância (ANOVA) que chama o valor de teste *F* (teste *F* formulado pelo BioEstat 5.0), complementada com o exame, a priori (Bonferroni), das diferenças entre as médias amostrais. Quando o valor de *F* é significativo e a escolha do teste das diferenças entre as médias for de Bonferroni, deve ser escolhido previamente o nível alfa, ou seja, como o teste é bilateral temos ($\alpha/2$) e escolhemos alfa igual a 0.025.

Em nossos experimentos, alguns parâmetros foram estabelecidos para todos os conjuntos de dados (Tabela 1), de acordo com [2, 11, 12]. A distância máxima busca l_{max} foi dada como segue:

$$l_{max} = \|U - L\| = \sqrt{\sum_{i=1}^{n} (U_i - L_i)^2} \quad (15)$$

Onde L_i e U_i são os limites inferiores e superiores para dimensão *i*th.

Foi testado para cada conjunto diferente de dados, número máximo de iterações para o GSO igual a 50, usando o melhor valor global (tabela 3). Para LM, o número máximo de iterações foi definido igual a 200 e o tamanho do grupo foi definido igual a 50 [2]. Vale ressaltar que estes parâmetros foram escolhidos baseados em testes feitos com outros valores.

Tabela 3 – Parâmetros fixos para todos os algoritmos

Algoritmo	**Parâmetro**	**Valor**
GSO	Porcentagem dos dependentes	80%
	θ_{max}	π/a^2
	αmax	$\theta_{max}/2$
	Número máximo de iterações	50
LM	Número de nós escondidos	6
	Número máximo de iterações	200
Decaimento de Pesos	Λ	5 X 10-6
	INC	1 X 10-3

Cada conjunto de dados foi dividido em conjuntos de validação, treinamento e testes, conforme especificado na Tabela 4. Para todos os algoritmos, 50 execuções independentes foram feitas com cada conjunto de dados. Os conjuntos de validação, treinamento e testes foram gerados aleatoriamente a cada tentativa de simulações. Os resultados obtidos pela LM, GSO e a abordagem do GSO cooperativo que alcançaram os melhores resultados estão em negrito.

Tabela 4 – Especificações das bases de dados

Bases de Dados	**Câncer**	**Diabetes**	**Ecoli**	**Vidros**
Treinamento	350	384	180	114
Validação	175	192	78	50
Teste	174	192	78	50

Na Tabela 5 a Tabela 8, os resultados obtidos para cada método são mostrados. A Tabela 5 mostra os resultados obtidos para a base câncer. O melhor resultado foi obtido pelo CGSO-H_k-WD (96,40% de *Accuracy* de teste) em uma avaliação empírica, e todas as abordagens variantes do GSO superaram o GSO tradicional e o algoritmo LM. A análise de variância (95% intervalo de confiança) mostrou que CGSO-H_k-WD e CGSO-S_k-WD alcançaram melhores resultados do que o algoritmo GSO tradicional e algoritmo LM. Apesar do tempo alcançado por eles ser um pouco maior do que tempo atingido pelas outras abordagens.

Tabela 5 – Resultados para Câncer

Algoritmos	**Tempo de Treinamento (s)**	***Accuracy* de Teste (%)**
LM	0,28376	87,32 ± 15,22
GSO	1041,38	95,68 ± 1,44
CGSO-WD	1043,84	96,08 ± 1,41
CGSO-S_k-WD	1262,58	96,33 ± 1,67
CGSO-H_k-WD	1249,74	**96,40 ± 1,46**

Para a base de diabetes (Tabela 6), em uma análise empírica, o CGSO-S_k-WD (76,42 de *Accuracy* de teste) superou todos os outros algoritmos, e todas as abordagens híbridas foram superiores ao GSO tradicional. O teste ANOVA mostrou que o algoritmo CGSO-S_k-WD foi melhor do que o algoritmo GSO tradicional e o algoritmo LM.

Tabela 6 – Resultados para Diabetes

Algoritmos	**Tempo de Treinamento (s)**	***Accuracy* de Teste (%)**
LM	0,31474	70,61 ± 7,99
GSO	1205,83	75,59 ± 2,39
CGSO-WD	1246,65	75,61 ± 2,80
CGSO-S_k-WD	1368,50	**76,42 ± 2,38**
CGSO-H_k-WD	1585,08	76,23 ± 2,56

A tabela 7, para a classificação de tipos de vidros, o melhor resultado empírico foi atingido por GSO-WD (64,24). Dois algoritmos baseado no GSO superaram o GSO tradicional e LM. O teste ANOVA mostrou que o GSO-WD foi melhor do que o algoritmo GSO tradicional e o algoritmo LM.

Tabela 7 – Resultados para Vidros

Algoritmos	**Tempo de Treinamento (s)**	***Accuracy* de Teste (%)**
LM	0,50688	45,96 ± 16,80
GSO	1917,85	61,96 ± 6,17
CGSO-WD	1872,31	**64,24 ± 7,36**
CGSO-S_k-WD	2343,07	61,52 ± 8,07
CGSO-H_k-WD	2389,07	62,44 ± 7,03

Em relação ao conjunto das bactérias Escherichia coli, conhecida como Ecoli, o GSO-WD (82,74) obtive maior *Accuracy* de teste para uma análise empírica. Alguns dos algoritmos baseados no GSO superaram o GSO tradicional e LM empiricamente, mas de acordo com teste ANOVA seus resultados são semelhantes entre si. O GSO-H_k-WD apesar de apresentar o valor da *accuracy* de teste maior do que o GSO tradicional, não apresenta um tempo de treinamento melhor do que o GSO tradicional.

Tabela 8 – Resultados para Ecoli

Algoritmos	**Tempo de Treinamento (s)**	***Accuracy* de Teste (%)**
LM	1,1889	53,64 ± 23,50
GSO	2362,94	82,64 ± 3,52
CGSO-WD	2841,36	81,82 ± 7,12
CGSO-S_k-WD	3467,36	82,00 ± 5,77
CGSO-H_k-WD	3604,1	**82,74 ± 4,70**

Os resultados mostram que o CGSO-H_k-WD obteve os melhore resultados entre os métodos avaliados para dois conjuntos de dados testados, o CGSO-H_k-WD para uma base e o CGSO-WD também para uma base. As abordagens baseadas no GSO obtiveram melhor desempenho do que GSO tradicional, apesar desta melhora não ter sido estatisticamente confirmada para a base vidros. De modo geral a melhora no desempenho não é tão significativa, porém para casos como diagnóstico de doenças, essa melhora pode significar grandes ganhos, por isso acreditamos ser válido o uso desta abordagem híbrida para otimizar as redes neurais para serem usadas na classificação destas bases.

5 Conclusão

Neste artigo, apresentamos uma nova abordagem de aprendizagem baseada na hibridização do algoritmo LM com o método GSO baseado na cooperação, nomeadamente CGSO. O CGSO usa a mecânica dos mínimos quadrados para determinar

analiticamente os pesos de saída e o algoritmo GSO para otimizar a entrada de pesos e bias ocultas. O desempenho dos algoritmos foi avaliado com conjuntos de dados de classificação clássicos do UCI *Machine Learning Repository*. Os resultados experimentais mostram que as abordagens híbridas propostas obtiveram melhor generalização do que o GSO original e o algoritmo LM para todos os conjuntos de dados testados. Para a maioria dos casos, as técnicas baseadas no GSO superaram o GSO tradicional e LM, de acordo com o teste de avaliação ANOVA. Em trabalhos futuros podemos considerar outras bases de dados e diversificar os tipos de problemas, bem como inserir novos nodos na camada escondida durante o processo de treinamento.

6 Referência

[1] C. J. Barnard and R. M. Sibly, "Producers and Scroungers: A General Model and Its Application to Captive Flocks of House Sparrows", in: Animal Behaviour, vol. 29, pp. 543-550, 1981.

[2] S. He, H. Wu and J. R. Saunders, "Group Search Optimizer: An Optimization Algorithm Inspired by Animal Searching Behaviour", in: IEEE Transactions on Evolutionary Computation, vol. 13, no. 5, pp. 973-990, 2009.

[3] Q. H. Wu, Z. Lu, M. S. Li and T. Y. Ji, "Optimal Placement of FACTS Devices by a Group Search Optimizer with Multiple Producers", in: 2008 Congress on Evolutionary Computation (CEC 2008), Hong Kong, pp. 1033-1039, 2008.

[4] S. He, Q. H. Wu and J. R. Saunders, "Breast Cancer Diagnosis Using an Artificial Neural Network Trained by Group Search Optimizer", in: Transactions of the Institute of Measurement and Control, vol. 31, no. 6, pp. 517-531, 2009.

[5] S. He and X. Li, "Application of a Group Search Optimization based Artificial Neural Network to Machine Condition Monitoring", in: 13th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2008), pp. 1260-1266, 2008.

[6] C. L. Higgins and R. E. Strauss, "Discrimination and classification of foraging path produced by search-tactic models," Behavior Ecology, vol. 15, pp. 248–254, 2003.

[7] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", in: Proc. IEEE Intl. Conf. on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NY, IV: 1942-1948, 1995.

[8] J. Kennedy and R. Eberhart. "Swarm Intelligence", Morgan Kaufmann Publishers, Inc, San Francisco, CA, 2001.

[9] P. N. Suganthan, "Particle Swarm Optimizer with Neighborhood Operator", in: Proc.Congr. Evolutionary Computation, Washington, DC, pp. 1958-1961, 1999.

[10] F. van den Bergh. "An Analysis of Particle Swarm Optimizers", PhD dissertation, Faculty of Natural and Agricultural Sciences, Univ. Pretoria, Pretoria, South Africa, 2002.

[11] F. van den Bergh and A. P. Engelbrecht, "A Cooperative Approach to Particle Swarm Optimization", IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pp. 225-239, 2004.

[12] M. Carvalho, T. B. Ludermir, "An Analysis of PSO Hybrid Algorithms for Feed-Forward Neural Networks Training", Proceedings of the Ninth Brazilian Symposium on Neural Networks – SBRN'06, pp. 6-11, 2006.

[13] M. Carvalho, T. B. Ludermir, "Particle Swarm Optimization of Feed-Foward Neural Networks with Weight Decay", in: Proceedings of the Sixth International Conference on Hybrid Intelligent Systems (HIS'06), pp. 5-5, 2006.

[14] S. Haykin, "Neural Networks: A Comprehensive Foundation", 2nd Edition, Prentice Hall, 1998.

[15] G. E. Hinton, "Learning Distributed Representation of Concepts", in: Annual Conference of Cognitive Science Society, pp. 1-12, 1986.

[16] A. S. Weigend, D. E. Rumelhart and B. A. Huberman, "Generalization by Weight Elimination with Application to Forecasting", in: Proceedings of the 1990 conference on Advances in neural information processing systems III (NIPS III), pp. 875-882, 1990.

[17] C. Zanchettin and T. B. Ludermir, "Global Optimization Methods for Designing and Training Feedforward Artificial Neural Networks", in: Dynamics of Continuous, Discrete & Impulsive Systems (DCDIS) A Supplement, Advances in Neural Networks, vol. 14 (S1), pp. 328-337, 2007.

[18] M. A. Potter and A. de Jong, "A Cooperative Coevolutionary Approach to Function Optimization", in: The Third Parallel Problem Solving from Nature, Berlin, Germany:Springer-Verlag, pp. 249-257, 1994.

[19] Y. Ong, A. Keane and P. Nair, "Surrogate-Assisted Coevolutionary Search", in: Proceeding of the 9th International Conference on Neural Information Processing, pp. 1140-1145, 2002.

[20] E. Cantú-Paz, "Efficient and Accurate Parallel Genetic Algorithms", 2nd Edition, Book Series on Genetic Algorithms and Evolutionary Computation, Norwell, MA:Kluwer, vol. I, 2000.

[21] Y. Xu, and Y. Shu, "Evolutionary Extreme Learning Machine – Based on Particle Swarm Optimization", Advances in Neural Networks – ISNN 2006, in: Lecture Notes in Computer Science, vol. 3971, pp. 644-652, 2006.

[22] A. Frank and A. Asuncion, UCI Machine Learning Repository, Univ. California, Sch. Inform. Comput. Sci., Irvine, CA, 2011 [Online]. Available: http://archive.ics.uci.edu/ml.