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

EXTREME LEARNING MACHINE BASEADA EM PSO COOPERATIVOS

D. N. G. Silva, L.D.S. Pacifico e T. B. Ludermir

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

Resumo – Extreme Learning Machine foi proposta como uma nova classe de algoritmo de aprendizagem para uma única camada escondida de uma rede neural feedforward (SLFN, do inglês single-hidden layer feedforward neural network) muito mais rápida do que a estratégia de aprendizagem tradicional baseada no gradiente. No entanto, a determinação aleatória dos pesos de entrada e bias ocultas da ELM pode levar a um desempenho não ideal, e poderá sofrer overfitting. Neste artigo são propostas quatro abordagens híbridas: GCPSO-ELM, CPSO-S_k-ELM, CPSO-H_k-ELM e GC-CPSO-H_k-ELM. Todas as abordagens são baseadas na estratégia de Otimização por Exame de Partícula (PSO, do inglês Particle Swarm Optimizer) a fim de otimizar a seleção de pesos de entrada e bias ocultas para o algoritmo ELM. Os resultados experimentais mostram que essas abordagens são capazes de conseguir um melhor desempenho do que a generalização da ELM tradicional.

Palavras Chave – Computação Evolucionária, Extreme Learning Machine, Otimização por Exame de Partícula, Sistemas Híbridos, Treinamento de Redes Neurais Artificiais.

1 Introducão

Extreme Machine Learning (ELM) foi proposta como uma nova classe de algoritmo de aprendizagem para uma única camada escondida de uma rede neural feedforward (SLFN) [1]. Amplamente usado, os algoritmos de aprendizagem baseados no gradiente, tais como back-propagation (BP) e variantes, são relativamente lentos na aprendizagem, mas a estratégia ELM, diferente do convencional, aumenta a velocidade de aprendizagem por gerar aleatoriamente os pesos de entrada e bias para nós ocultos em vez de iterativamente ajustar os parâmetros de rede. Embora a ELM seja rápida e apresente boa performance de generalização, visto que os pesos de saída são calculados com base nos pesos de entrada prefixados e bias ocultas usando a inversa generalizada Moore-Penrose (MP), pode existir um conjunto de pesos de entrada e bias escondidos não ideal ou desnecessários. A ELM pode exigir mais neurônios escondidos do que os algoritmos convencionais de regularização de aprendizagem baseados em algumas aplicações, o que pode tornar a ELM lenta aos dados de teste desconhecido [2].

Técnicas de pesquisa global, com a possibilidade de alargar o espaço de busca, na tentativa de evitar mínimos locais, têm sido amplamente utilizadas para ajuste de pesos de conexão ou otimização da arquitetura em redes neurais, tais como Algoritmos Evolutivos (AE) [10], Simulated Annealing (SA) [11], Tabu Search (TS) [12], Otimização por Colônia de Formigas (ACO - Colony Optimization Algorithm) [13] e Otimização por Enxame de Partículas (PSO - Particle Swarm Optimization) [4, 5]. Desde a criação do PSO, diversas melhorias têm sido propostas, tais como o uso da cooperação entre as populações isoladas [7, 9]. Uma das primeiras formas de cooperação foi estudada com base na evolução de ilhas [10], onde cada ilha corresponde a uma sub-população isolada geograficamente pesquisando em uma área separada do espaço de solução. Depois de *m* interações, as ilhas enviam e recebem $1 \leq z \leq 5$ indivíduos que promovem um intercâmbio de informações entre as ilhas. As interações entre as ilhas são regidas pela topologia da comunicação. As topologias mais utilizadas são as de anel, estrela, malha, entre outras. Outra forma de cooperação que tem obtido bons resultados em problemas de otimização numérica é a baseada na divisão do espaço de busca, que é aplicada a problemas de alta dimensionalidade. Assim, o espaço de busca original de dimensão *n* é dividido em $1 \leq k \leq n$ partições de dimensão *d*, com $k \times d = n$. No PSO esta abordagem foi introduzida pelo algoritmo CPSO-S_k [7, 9].

Algumas estratégias evolutivas têm sido adotadas para o contexto da ELM. Zhu *et al*. [2] apresenta uma forma híbrida do algoritmo Evolução Diferencial (DE) e ELM, chamado E-ELM para treinar SLFN com redes mais compactas. Xu e Shu [3] apresentaram um novo ELM evolutivo baseado em PSO para a tarefa de previsão. Em Sundararajan *et al*. [8], uma combinação de Integer Coded Algoritmo Genético (ICGA) e PSO, juntamente com a ELM tem sido utilizado para a seleção de genes e classificação do câncer, onde o ICGA e PSO-ELM selecionam um conjunto ideal de genes que são então usados para construir um classificador para desenvolver um algoritmo (ICGA_PSO_ELM) que pudesse lidar com dados esparsos e desequilíbrio da amostra.

Neste trabalho, nós apresentamos quatro novas abordagens híbridas que tem as vantagens do algoritmo ELM e algumas variações do algoritmo Particle Swarm Optimization (PSO): a Convergência Garantida (GCPSO-ELM), o PSO Cooperativo (CPSO-S_k-ELM), o CPSO Híbrido (CPSO-H_K-ELM) e uma versão híbrida que combina convergência garantida e o CPSO híbrido (chamado GC-CPSO-H_K-ELM). Os experimentos foram feitos com os algoritmos ELM e PSO-ELM [3], e as quatro abordagens introduzidas usando problemas reais de classificação clássicos (Câncer, Diabetes, Ecoli, Vidros, Coração e Iris), provenientes do UCI Machine Learning Repository [14].

Este artigo está organizado da seguinte forma. A Seção II apresenta o ELM padrão [1], o PSO padrão [4, 5], o PSO com convergência garantida (GCPSO) e dois PSO baseados em abordagens cooperativas: o PSO Cooperativo (CPSO-S_k) [7, 9] e o

CPSO Híbrido (CPSO-H_K) [7]. Em seguida, as nossas abordagens são apresentadas na seção III e os resultados experimentais são mostrados na seção IV. As conclusões são dadas na seção V.

2 Preliminares

2.1. Extreme Learning Machine (ELM)

Extreme Learning Machine (ELM) foi proposta por Huang, *et al*. [1]. Parte-se do pressuposto que estamos treinando SLFNs com K neurônios escondidos e função de ativação $g(x)$ para aprender N amostras distintas $(\boldsymbol{x}_i, \boldsymbol{t}_i)$, onde $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in \mathbb{R}^n$ e $\boldsymbol{t}_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in \mathbb{R}^m$. Em ELM, os pesos de entrada e bias ocultas são gerados aleatoriamente, em vez de iterativamente ajustar os parâmetros de rede. Ao fazer isso, o sistema não linear é convertido para um sistema linear:

$$\boldsymbol{H}\beta = T$$

Onde $\boldsymbol{H} = \{h_{ij}\}$ ($i = 1,\ldots, N$ e $j = 1, \ldots, K$) é a saída da camada escondida da matriz, $h_{ij} = g(w_j \cdot x_i + b_j)$ indica a saída do neurônio oculto jth no que diz respeito a x_i; $w_j = [w_{j1}, w_{j2},\ldots, w_{jn}]^T$ é o vetor de pesos das conexões dos neurônios ocultos jth e com os neurônios de entrada, e b_j denota a bias do jth neurônio oculto; $w_j \cdot x_i$ denota o produtor interno do w_j e x_i ; $\beta = [\beta_1, \beta_2,\ldots, \beta_K]^T$ é a matriz pesos de saída e $\beta_j = [\beta_{j1}, \beta_{j2},\ldots, \beta_{jm}]^T$ ($j = 1, \ldots, K$) denota o vetor de pesos conectando o jth neurônios ocultos e neurônios de saída; $T = [\boldsymbol{t}_1, \boldsymbol{t}_2,\ldots, \boldsymbol{t}_n]^T$ é a matriz de metas (saída desejada).

Assim, a determinação dos pesos de saída (que liga a camada escondida para a camada de saída) é tão simples como encontrar a solução de mínimos quadrados para o dado sistema linear. A norma mínima dos mínimos quadrados (LS - least-square) da solução para o sistema linear é

$$\hat{\beta} = \boldsymbol{H}^{\dagger} T$$

Onde $\boldsymbol{H}^{\dagger}$ é a inversa generalizada MP da matriz $\boldsymbol{H}$. A norma mínima dos mínimos quadrados da solução é única e tem a menor norma entre todas as soluções LS. Conforme analisado por Huang *et al*. [1], ELM usando o método MP tende a obter um bom desempenho de generalização com o aumento na velocidade de aprendizagem.

2.2. Mantendo a Integridade das Especificações

A técnica de otimização PSO foi introduzida por Kennedy e Eberhart, em [4] como uma busca estocástica por meio de um problema de espaço n-dimensional tendo em vista a minimização (ou maximização) da função objetivo do problema. O PSO foi construído através da tentativa de simular graficamente a coreografia de um bando de pássaros que voam em busca de recursos. Mais tarde, à procura de fundamentos teóricos, estudos foram realizados sobre a maneira como os indivíduos interagem em grupos, trocando informações e revendo conceitos pessoais para melhorar a sua adaptação ao ambiente [5].

Em PSO, um enxame de soluções (partículas) é mantido. Cada partícula individual mantém a sua posição, velocidade e melhor posição. Como o algoritmo itera, a velocidade de cada partícula é determinada de acordo com os dois principais pontos de referência da pesquisa: a melhor posição individual visitada até agora $y_i(t)$ e a melhor posição global visitada até agora $\hat{y}(t)$. As equações (1) e (2) descrevem, respectivamente, como a nova velocidade e a nova posição de uma partícula são determinadas.

$$v_{ij}(t+1) = w v_{ij}(t) + c_1 r_1 (y_{ij}(t) - x_{ij}(t)) + c_2 r_2 (\hat{y}_j(t) - x_{ij}(t)) \qquad 1 \le i \le s,\ 1 \le j \le n \tag{1}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \qquad 1 \le i \le s,\quad 1 \le j \le n \tag{2}$$

Onde w é o peso de inércia escalar (termo momentum geralmente no intervalo [0.4, 0.9]), os valores r_1 and r_2 são variáveis aleatórias retiradas de uma distribuição uniforme U (0, 1), e os valores $0 < c_1, c_2 \le 2$ são coeficientes de aceleração individual e global, respectivamente, normalmente definido para valores iguais, mas as vezes valores diferentes para c_1 e c_2 leva a um melhor desempenho [6]. A melhor posição individual visitada até agora $y_i(t)$ é atualizada de acordo com a equação (3), enquanto que a melhor posição global visitada até agora $\hat{y}(t)$ é atualizada por meio da equação (4).

$$y_i(t+1) = \begin{cases} y_i(t), & \text{if } f(x_i(t+1)) \ge f(y_i(t)) \\ x_i(t+1), & \text{if } f(x_i(t+1)) < f(y_i(t)) \end{cases} \tag{3}$$

$$\hat{y}(t+1) = \arg\min_{yi(t+1)} f(y_i(t+1)), \qquad 1 \le i \le s \tag{4}$$

2.3. PSO com Convergência Garantida (GCPSO)

O PSO padrão tem uma propriedade que se $x_i = y_i = \hat{y}_i$, o que significa que a partícula *i* está situada no melhor ponto do espaço de busca encontrado até agora, então a equação de atualização da velocidade (equação 1) é totalmente dependente do termo de inércia $wv_i(t)$. Se a velocidade anterior da partícula é muito próxima a zero, em seguida, a partícula vai parar de se mover, empurrando as partículas a esse ponto e causando a convergência prematura do enxame. Uma pequena modificação no PSO padrão é feita pelo algoritmo de convergência garantida (GCPSO) [9] para lidar com este problema. A idéia é modificar a equação de atualização da velocidade só para as partículas que atingiram o ponto mais global do espaço de busca para evitar a convergência prematura do enxame e, ao mesmo tempo, procurar as melhores soluções nos arredores da atual melhor posição global $\hat{y}$. A nova equação utilizada é representada pela equação (5) em que *i* é o índice de uma partícula que alcançou a melhor posição atual do enxame e $r(t)$ é um número aleatório uniforme tomado a partir de $U(0, 1)$. As outras partículas do enxame continuam a usar a equação de atualização padrão da velocidade, ou seja, a equação (1).

$$v_{ij}(t+1) = -x_{ij}(t) + \hat{y}_j(t) + wv_{ij}(t) + \rho(t)(1-2r(t)) \qquad (5)$$

O termo $\rho(t)$ da equação é um fator de escala de adaptação que faz o PSO executar uma busca aleatória em torno da partícula do enxame. A próximo valor $\rho(t)$ é determinado pela equação (6), em que *#sucessos* e *#fracassos* denotados pelo número de sucessos e fracassos consecutivos da busca em minimizar a função objetivo, e s_c e f_c são parâmetros de limites com valores iniciais geralmente iguais a 5. Sempre que o contador dos *#sucessos* ultrapassa o limiar de sucesso, significa que a área circundante a melhor posição poderá ser ampliada levando à duplicação do valor $\rho(t)$. Da mesma forma, quando o contador de *#fracassos* exceder o limite de falhas, isso significa que a área circundante a melhor posição global é muito grande e precisa ser reduzida, como pode ser visto na equação (6).

$$\rho(t+1) = \begin{cases} 2\rho(t), & if\ \#sucessos \geq s_c \\ 0.5\rho(t), & if\ \#fracassos \geq f_c \\ \rho(t), & senão \end{cases} \qquad (6)$$

Toda vez que os contadores de sucessos ou fracassos ultrapassarem seus limites correspondentes, s_c e f_c, respectivamente, o limiar excedido é aumentado.

A cada iteração que a busca consegue minimizar a melhor posição atual, o contador de *#sucessos* é incrementado e contador *#fracassos* é reposto a zero. Da mesma forma, a cada iteração que a melhor posição global $\hat{y}(t)$ não é atualizada, o contador de *#fracassos* é incrementado e o contador de *#sucessos* é reduzido a zero.

2.4. PSO Cooperativo (CPSO-S_k)

Desde a criação do PSO, diversas melhorias têm sido propostas. Uma delas é que o PSO faz uso da cooperação entre as populações. Embora a concorrência entre os indivíduos geralmente resulte em um bom desempenho, novas melhorias podem ser obtidas pelo uso da cooperação entre os indivíduos e, além disso, entre as populações isoladas [7, 9]. Uma forma de cooperação que tem obtido bons resultados em problemas de otimização numérica é baseada na divisão do espaço de busca, que é aplicada a problemas de alta dimensionalidade. Assim, o espaço de busca original de dimensão *n* é dividido em $1 \leq k \leq n$ partições de dimensão *d*, com $k \times d = n$. No PSO esta abordagem foi introduzida pelo algoritmo CPSO-S_k [7, 9].

Embora o espaço de busca de dimensão *n* tenha sido dividido em *k* partições de dimensão *d* em que uma sub-pesquisa é executada, o problema permanece como um *n*-dimensional. Assim, o sub-população de dimensão *d* necessita cooperar oferecendo o seu melhor sub-indivíduo para completar as informações necessárias para a avaliação da função objetivo $f: \mathbb{R}^n \rightarrow \mathbb{R}$. Formalmente, a cooperação entre as sub-populações é feita através da concatenação dos atuais sub-indivíduos que desejam avaliar e os melhores sub-indivíduos tão longe das demais partições, nas posições correspondentes. Esta composição é representada pelo vetor contexto resultante da função b(j, *vec*), expresso na equação (7), com *vec* como o sub-individuo da sub-população *j* que queremos avaliar e $Pi.\hat{y}$ como o melhor sub-indivíduo da sub-população *i*.

$$b(j, vec) \equiv (P_1.\hat{y}, P_2.\hat{y}, ..., P_{j-1}.\hat{y}, vec, P_{j+1}.\hat{y}, ...P_k.\hat{y}) \qquad (7)$$

Assim, podemos descrever o algoritmo CPSO-S_k como tendo realizado a cooperação de PSOs que otimiza cada uma das *k* partições do espaço de busca. O termo *k* é o fator de partição da dimensão do problema.

2.5. CPSO Híbrido (CPSO-H_k)

O CPSO Híbrido (CPSO-H_k) [7], combina a capacidade de escapar de pseudo-mínimos do PSO padrão com convergência mais rápida em determinadas funções alcançadas por CPSO-S_k, de tal forma que o algoritmo CPSO-S_k é executado por uma iteração, seguido de uma iteração do algoritmo PSO. São trocadas (cooperação) informações sobre as melhores soluções encontradas até agora por qualquer um dos componentes ao final de cada iteração.

Um mecanismo simples para implementar esta troca de informações é a substituição de algumas das partículas de metade do algoritmo com a melhor solução descoberta até agora pela outra metade do algoritmo. Especificamente, após uma iteração da metade do algoritmo CPSO-S_k (o enxame P_j), o vetor contexto $b(1,P1.\hat{y})$ é usado para substituir uma partícula escolhidas

aleatoriamente, na metade do algoritmo PSO (o enxame Q). Isso é seguido por uma iteração do enxame Q do componente do algoritmo, que produz uma nova melhor partícula global, $Q.\hat{y}$. Esse vetor é então dividido em sub-vetores de dimensão apropriada e utilizado para substituir as posições das partículas escolhidas aleatoriamente no enxame P_j.

Embora as partículas que são substituídas durante o processo de troca de informações são escolhidas aleatoriamente, o algoritmo não substitui a posição do melhor global de qualquer um dos enxames, já que este poderia ter um efeito negativo sobre o desempenho do enxame afetado.

3 Extreme Learning Machine Híbrido Evolucionário

Esta seção apresenta quatro novas abordagens híbridas baseadas no algoritmo ELM e no PSO com Convergência Garantida e suas variantes cooperativas: GCPSO-ELM, CPSO-S_k-ELM, CPSO-H_k-ELM e GC-CPSO-H_k-ELM. Todas as abordagens tentaram encontrar o melhor conjunto de pesos de entrada e bias ocultas usando variantes do PSO, de tal maneira que foi necessário um número reduzido de nós escondidos para alcançar um bom desempenho da rede. Como explicado na seção II, convergência garantida impede o PSO da convergência prematura. As variações cooperativas do PSO adotadas baseiam- se na divisão do espaço de busca em k enxames de modo que cada enxame tenta encontrar a melhor solução para um número específico da dimensão do problema, combinando suas melhores soluções com as soluções obtidas por enxames de outros para resolver todo o problema. O algoritmo GC-CPSO- H_k-ELM usa as vantagens obtidas por ambas abordagens, do PSO híbrido cooperativo (CPSO-H_k) e a do PSO com Convergência Garantida (GCPSO). Essa combinação foi feita de tal forma que a Convergência Garantida foi aplicada somente no PSO padrão na etapa do CPSO-H_k (População Q na Figura 3).

A inversa generalizada MP é usada para obter os pesos de saída, como na tradicional ELM [1].

Para todos os algoritmos, o enxame inicial é gerado aleatoriamente. Cada partícula i no enxame é composta por um conjunto de pesos de entrada e bias ocultas:

$$x_i = [w_{11}, w_{12}, \ldots, w_{1K}, w_{21}, w_{22}, \ldots, w_{n1}, w_{n2}, \ldots w_{nK}, \ldots, b_1, b_2, \ldots, b_K]$$

Todos w_{ij} e b_j são inicializados aleatoriamente dentro do intervalo de [-1, 1]. Para cada indivíduo, os pesos de saída correspondente da matriz são calculados usando a inversa generalizada MP. A função de fitness adotada é a raiz do erro médio quadrado (RMSE) do conjunto de validação [2]:

$$E = \sqrt{\frac{\sum_{j=1}^{N} || \sum_{i=1}^{K} \beta_i g(w_i . x_j + b_i) ||_2^2}{m \; x \; N}} \qquad (8)$$

Os Pseudocódigos para GCPSO-ELM, CPSO-S_k-ELM e CPSO-H_k-ELM são dados nas figuras de 1 a 3. Depois que a população inicial é gerada, cada variante do PSO (GCPSO-ELM, CPSO-S_k-ELM, CPSO-H_k-ELM e GC-CPSO-H_k-ELM) executa seus passos, usando ELM como a função de fitness. O pseudocódigo do GC-CPSO-H_k-ELM pode ser construído com base na figura 3 utilizando os passos da figura 1 onde é feita a modificação da equação de atualização da velocidade só para as partículas que atingiram o ponto mais global do espaço de busca.

1. inicialize randomicamente a população de partículas
2. **repita**
3. **para** cada partícula i da população **faça**
4. avalie $f(x_i)$ com o RMSE obtido pelo conjunto de validação da ELM
5. **se** $f(x_i(t)) < f(y_i(t))$ **então**
6. $y_i(t) = x_i(t)$
7. **fim do se**
8. **se** $f(y_i(t)) < f(\hat{y}(t))$ **então**
9. $\hat{y}(t) = y_i(t)$
10. **fim do se**
11. **fim do para**
12. atualização da velocidade e posição de cada partícula de acordo com as equações. (1) e (2). Para a melhor partícula encontrada até o momento, a velocidade de atualização será de acordo com a equação (5)
13. atualização de ρ de acordo com a equação. (6), e *#sucessos*, *#fracassos*, s_c e f_c, se necessário
14. **até** o critério de parada ser satisfeito

Figura 1 – Pseudocódigo para o algoritmo GCPSO-ELM

Uma estratégia de delimitar o espaço de pesquisa foi adotada para resolver o problema das partículas que tentam escapar das dimensões do problema. Assim, quando uma partícula foge do espaço de busca, ela é forçada a voltar para o espaço de pesquisa, definindo as variáveis que violaram os limites de seus valores anteriores. Algumas estratégias diferentes são mostradas em Xu e Shu [3].

1. **defina**
2. $b(j, z) \equiv (P_1.\hat{y}, \ldots, P_{j-1}.\hat{y}, z, P_{j+1}.\hat{y}, \ldots, P_k.\hat{y})$
3. $K_1 = n \bmod K$ *e* $K_2 = K\text{-}1 \; (n \bmod K)$

4. Inicialize $K_1 \lceil n \setminus K \rceil$-dimensional PSOs: $P_{1,} j \in K$ [*1*… K_1]
5. Inicialize $K_2 \lfloor n \setminus K \rfloor$-dimensional PSOs: $P_{1,} j \in K$ [(K_1+1)… K]
6. **repita**:
7. **para** cada enxame $j \in [1 \ldots K]$:
8. **para** cada partícula $i \in [1 \ldots s]$:
9. avalie $f(b(j, P_i.x_i))$ com o RMSE obtido pelo conjunto de validação da ELM
10. **se** $f(b(j, P_j.x_i)) < f(b(j, P_j.y_i))$
11. **então** $P_j.y_i = P_j.x_i$
12. **se** $f(b(j, P_j.y_i)) < f(b(j, P_j.\hat{y}))$
13. **então** $P_j.\hat{y} = P_j.y_i$
14. **fim do para**
15. performance de atualização do PSO em P_j usando (1-2)
16. **fim do para**
17. **até** o critério de parada ser satisfeito

Figura 2 – Pseudocódigo para o algoritmo CPSO-S_k-ELM

1. defina $b(\boldsymbol{j}, z) \equiv (\boldsymbol{P_1.\hat{y}, \ldots, P_{j-1}.\hat{y}, z, P_{j+1}.\hat{y}, \ldots, P_k.\hat{y}})$
2. $K_1 = n \bmod K$ e $K_2 = K\text{-}1$ ($n \bmod K$)
3. Inicialize $K_1 \lceil n \setminus K \rceil$-dimensional PSOs: $P_{1,} j \in K$ [*1*… K_1]
4. Inicialize $K_2 \lfloor n \setminus K \rfloor$-dimensional PSOs: $P_{1,} j \in K$ [(K_1+1)… K]
5. Inicialize um n-dimensional PSO: Q
6. **repita:**
7. **para** cada enxame $\boldsymbol{j \in [1 \ldots K]}$:
8. **para** cada partícula $\boldsymbol{i \in [1 \ldots s]}$:
9. avalie $f(b(j, P_i.x_i))$ com o RMSE obtido pelo conjunto de validação da ELM
10. **se** $f(b(j, P_j.x_i)) < f(b(j, P_j.y_i))$
11. **então** $P_j.y_i = P_j.x_i$
12. **se** $f(b(j, P_j.y_i)) < f(b(j, P_j.\hat{y}))$
13. **então** $P_j.\hat{y} = P_j.y_i$
14. **fim do para**
15. Performance de atualização do PSO em P_j usando (1-2)
16. **fim do para**
17. Selecionar randomicamente $k \sim U(1, s/2 \mid Q.y_k \neq Q.\hat{y}$ *e* $Q.x_k = b(1, P_1, \hat{y})$
18. **para** cada partícula $i \in [1 \ldots s]$:
19. avalie $f(Q.x_i)$ com o RMSE obtido pelo conjunto de validação da ELM
20. **se** $f(Q.x_j) < f(Q.y_j)$
21. **então** $Q.y_j = Q.x_j$
22. **se** $f(Q.y_j) < f(Q.\hat{y})$
23. **então** $Q.\hat{y} = Q.y_j$
24. **fim do para**
25. Performance de atualização do PSO em Q usando (1-2)
26. **para** o enxame $j \in [1 \ldots K]$:
27. selecione randomicamente $k \sim U(1, s/2 \mid P_j.y_k \neq P_j.\hat{y}$
28. $P_j.\, x_k = Q.\, \hat{y}_j$
29. **fim do para**
30. **até** o critério de parada ser satisfeito

Figura 3 – Pseudocódigo para o algoritmo CPSO-H_k-ELM

4 Resultados e Discussão

Nesta seção, comparamos o ELM tradicional com o LM (uma das mais rápidas implementações do algoritmo back-propagation), PSO-ELM [3] e as suas novas variantes híbridas, ou seja as variantes cooperativas do PSO: GCPSO-ELM, CPSO-S_k-ELM, CPSO-H_k-ELM e GC-CPSO-H_k-ELM. Todos os experimentos foram executados no ambiente MATLAB 6,0, e o algoritmo LM é fornecido pela caixa de ferramentas (toolbox) de Redes Neurais do MATLAB. Um conjunto de validação é usado em LM, e em todas as variantes do PSO para evitar overfitting.

Para avaliar todos os algoritmos, seis conjuntos de dados de classificação de referência (câncer, diabetes, ecoli, vidros, coração e diafragma), obtido a partir da UCI Machine Learning Repository [14] têm sido utilizados. Estes conjuntos de dados apresentam diferentes graus de dificuldades, números de classes diferentes e números de exemplos e características diferentes. As métricas de avaliação utilizadas são a análise empírica e o teste de hipóteses de Wilcoxon. Em nossos experimentos, todas as entradas (atributos) foram normalizadas no intervalo [0, 1], enquanto as saídas (metas) foram normalizar em [-1, 1].

Os pesos de entrada e as bias foram obtidos no intervalo [-1, 1]. A ELM utilizou a função de ativação sigmóide $g(x) = 1 / (1 + exp(-x))$. Em nossos experimentos, alguns parâmetros foram estabelecidos para todos os conjuntos de dados (Tabela 1), de acordo com [7, 9]. Foram testados para cada conjunto de dados diferentes número máximo de iterações do PSO, número k de

enxames para o PSO cooperativo, utilizando os melhores valores encontrados. Para LM, o número máximo de épocas foi definido para 200.

O tamanho do enxame é definido como o dobro do número de parâmetros da rede [2]. O número de nós escondidos de ELM foi diferente para cada conjunto de dados e para cada algoritmo de acordo com os melhores resultados obtidos pela primeira etapa de avaliações.

Tabela 1 – Parâmetros fixos para todos os algoritmos

Algoritmos	**Parâmetros**	**Valor**
PSO	c_1	2.0
	c_2	2.0
	w	0.9 to 0.4
	Número máximo de iteração do PSO	50
GCPSO	ρ_0	1.0
	s_c	5
	f_c	5
CPSO-S_k	K	10

Cada conjunto de dados foi dividido em conjuntos de treinamento, validação e teste, conforme especificado na Tabela 2. Para todos os algoritmos, 50 execuções independentes foram feitas com cada conjunto de dados. Os conjuntos de treinamento, validação e teste foram gerados aleatoriamente a cada tentativa da simulação. O resultado obtido por ELM, PSO-ELM e pela abordagem híbrida que alcançou o melhor resultado entre todas as abordagens híbridas estão em negrito.

Tabela 2 – Especificações das bases de dados

Base de Dados	**Câncer**	**Diabetes**	**Ecoli**	**Vidros**	**Coração**	**Iris**
Classes	2	2	8	7	2	3
Atributos	9	8	7	9	13	4
Treinamento	350	252	180	114	130	70
Validação	175	258	78	50	70	40
Teste	174	258	78	50	70	40
Total	699	768	336	214	270	150

Nas de Tabela 3 a Tabela 8, são mostrados os resultados obtidos para cada método. A Tabela 3 mostra os resultados obtidos para a base de dados Câncer. O melhor resultado foi obtido pelo método híbrido GC-CPSO-H_k-ELM (96,85% de acerto), e em uma avaliação empírica e pelo teste de Wilcoxon (intervalo de confiança 95%) foi mostrado que todas as variantes do PSO (PSO-ELM, GCPSO-ELM, CPSO-S_k-ELM, CPSO-H_k-ELM e GC-CPSO-H_k-ELM) obtiveram melhores resultados do que os algoritmos tradicionais ELM e LM, mas são semelhantes entre si. O teste de Wilcoxon também mostrou que ELM superou o algoritmo LM para este conjunto de dados.

Tabela 3 – Resultados para Câncer

Algoritmo de Classificação	**Tempo de Treinamento (s)**	**Accuracy de Teste**	**Nós Ocultos**
ELM	0.0014	**96.09 ± 1.48**	10
PSO-ELM	32.31	**96.70 ± 1.05**	5
GCPSO-ELM	36.55	96.83 ± 1.01	5
CPSO-S_k-ELM	34.49	96.82 ± 1.12	5
CPSO-H_k-ELM	34.38	96.73 ± 1.00	5
GC-CPSO-H_k-ELM	35.74	**96.85 ± 1.06**	5
LM	0.45	91.71± 10.25	10

Para casos de diabetes (Tabela 4), em uma análise empírica o GPSO-ELM superou todos os outros algoritmos, e todas as abordagens cooperativas baseadas no PSO foram superiores a ELM tradicional, PSO-ELM e LM. O teste de Wilcoxon (intervalo de confiança 95%) mostrou que todos os algoritmos baseados no PSO (PSO-ELM, GCPSO-ELM, CPSO-S_k-ELM, CPSO-H_k-ELM e GC-CPSO-H_k-ELM) apresentaram um comportamento semelhante entre si e são melhores do que algoritmos ELM e LM. ELM tradicional e LM obtiveram desempenhos similares, segundo o teste de Wilcoxon, mas seus resultados são os piores empiricamente.

Tabela 4 – Resultados para Diabetes

Algoritmo de Classificação	**Tempo de Treinamento (s)**	**Accuracy de Teste**	**Nós Ocultos**
ELM	0.0014	**75.89 ± 2.47**	10
PSO-ELM	81.52	**76.50 ± 2.22**	10
GCPSO-ELM	77.02	**76.95 ± 2.49**	10

CPSO-S_k-ELM	82.08	76.67 ± 2.09	10
CPSO-H_k-ELM	70.88	76.73 ± 2.30	10
GC-CPSO-H_k-ELM	84.70	76.87 ± 2.28	10
LM	0.43	74.50 ± 4.25	10

Para o conjunto de dados das bactérias Escherichia coli, conhecida como Ecoli (Tabela 5), GC-CPSO-H_k-ELM obteve melhor resultado entre todas as abordagens na avaliação empírica, e só esta abordagem foi melhor do que o algoritmo tradicional ELM em relação ao teste de Wilcoxon (95% intervalo de confiança). Em uma análise empírica todas as outras abordagens cooperativas superaram o PSO-ELM e ELM, mas em relação ao teste de Wincoxon (95% intervalo de confiança), seus resultados são semelhantes. Novamente, o LM obteve o pior resultado, e mostrou o mais alto grau de instabilidade.

Tabela 5 – Resultados para Ecoli

Algoritmo de Classificação	**Tempo de Treinamento (s)**	**Accuracy de Teste**	**Nós Ocultos**
ELM	0.0023	**85.31** ± **3.30**	20
PSO-ELM	92.07	**85.72 ± 3.58**	20
GCPSO-ELM	109.32	86.13 ± 3.11	20
CPSO-S_k-ELM	94.75	86.23 ± 3.57	20
CPSO-H_k-ELM	90.86	86.23 ± 3.61	20
GC-CPSO-H_k-ELM	108.08	**86.44 ± 2.86**	20
LM	7.05	65.64± 21.43	20

Resultados obtidos para o conjunto de Vidros (Tabela 6) mostrou que o algoritmo CPSO-H_k-ELM obtece um desempenho melhor do que todas as outras abordagens de forma empírica, e melhor do que os algoritmos ELM, PSO-ELM e LM tradicional, em um teste de Wilcoxon (95% intervalo de confiança). GCPSO-ELM, CPSO-S_k-ELM e GC-CPSO-H_k-ELM são melhores do que o PSO-ELM em uma análise empírica, mas de acordo com o teste de Wilcoxon seu desempenho é semelhante. Todas essas abordagens são melhores do que ELM e algoritmos LM em ambas as avaliações, a empírica e pelo teste de Wilcoxon. LM obteve o pior resultado para este conjunto de dados e um alto grau de instabilidade.

Tabela 6 – Resultados para Vidros

Algoritmo de Classificação	**Tempo de Treinamento (s)**	**Accuracy de Teste**	**Nós Ocultos**
ELM	0.0018	**60.53 ± 5.35**	25
PSO-ELM	99.60	**62.17 ± 5.64**	25
GCPSO-ELM	105.57	62.35 ± 5.15	25
CPSO-S_k-ELM	101.19	62.42 ± 4.30	25
CPSO-H_k-ELM	100.81	**63.54 ± 5.02**	25
GC-CPSO-H_k-ELM	119.34	62.77 ± 4.78	25
LM	8.54	47.16 ± 15.39	25

A Tabela 7 mostra que para o conjunto de dados de Coração o GC-CPSO-H_k-ELM alcançou os melhores resultados empiricamente. Também foi melhor que os algoritmos ELM, PSO-ELM e LM de acordo com o teste de Wilcoxon (95% intervalo de confiança), mas seus resultados são semelhantes a todas as outras variantes do PSO de acordo com este teste. O teste de Wilcoxon mostrou também que ELM e LM obtiveram desempenho pior do que todos os algoritmos baseados no PSO cooperativo.

Tabela 7 – Resultados para Coração

Algoritmo de Classificação	**Tempo de Treinamento (s)**	**Accuracy de Teste**	**Nós Ocultos**
ELM	0.0018	**80.48** ± **4.87**	20
PSO-ELM	21.00	**81.20 ± 4.95**	5
GCPSO-ELM	19.06	81.91 ± 4.31	5
CPSO-S_k-ELM	19.07	81.60 ± 4.6	5
CPSO-H_k-ELM	26.30	82.26 ± 4.91	5
GC-CPSO-H_k-ELM	20.91	**82.48 ± 4.71**	5
LM	1.33	77.28 ± 6.92	20

Em relação ao conjunto de plantas Iris (Tabela 8), o melhor resultado empírico foi atingido por CG-CPSO-H_k-ELM, e todas as abordagens baseadas no PSO superaram os algoritmos tradicionais ELM e LM. O teste de Wilcoxon, com um intervalo de confiança de 95%, mostrou que PSO-ELM, CPSO-H_k-ELM e GC-CPSO-H_k-ELM superaram o ELM, mas GCPSO-ELM e CPSO-S_k-ELM realizaram uma performance semelhante ao ELM . Todas as abordagens superaram algoritmo LM tanto na análise empírica quanto no teste de Wilcoxon.

Tabela 8 – Resultados para Iris

Algoritmo de Classificação	Tempo de Treinamento (s)	Accuracy de Teste	Nós Ocultos
ELM	0.0013	**95.60 ± 3.70**	15
PSO-ELM	16.39	**96.60 ± 2.93**	15
GCPSO-ELM	16.89	96.60 ± 2.89	15
CPSO-S_k-ELM	17.30	96.25 ± 2.78	15
CPSO-H_k-ELM	20.88	96.85 ± 2.51	15
GC-CPSO-H_k-ELM	18.68	**97.15 ± 2.14**	15
LM	0.48	82.95 ± 2.23	15

Os resultados mostraram que GC-CPSO-H_k-ELM obteve melhores resultados entre todos os métodos avaliados em uma análise empírica para os conjuntos de dados testados. Esses resultados podem ser atribuídos ao fato de que a GC-CPSO-H_k-ELM combina de forma híbrida as melhores características de PSO com convergência garantida, PSO cooperativo (CPSO-S_k) e o algoritmo ELM.

5 Conclusão

Neste artigo, apresentamos quatro novos enfoques de aprendizagem baseados em dados de hibridização do algoritmo ELM com métodos PSO com convergência garantida e cooperativos, ou seja, algoritmos GCPSO-ELM, CPSO-S_k-ELM, CPSO-H_k-ELM e GC-CPSO-H_k-ELM. Eles usam regime da norma do mínimo quadrado para determinar analiticamente os pesos de saída e variantes do PSO para otimizar os pesos de entrada e bias ocultas.

O desempenho dos algoritmos foi avaliado com conjuntos de dados de classificação clássicos (Câncer, Diabetes, Ecoli, Vidros, Coração e Iris), obtido a partir da UCI Machine Learning Repository [14]. Os resultados experimentais mostraram que as nossas abordagens híbridas alcançar melhores desempenhos de generalização do ELM original para todos os conjuntos de dados testados. Os resultados também mostraram que o GC-CPSO-H_k-ELM apresentou melhor desempenho entre todos os algoritmos testados em uma análise empírica e para alguns casos pelo teste de Wilcoxon. Todos os resultados obtidos para as abordagens híbridas superaram o algoritmo Levenberg-Marquardt.

Como trabalhos futuros, pretendemos avaliar a influência de diferentes condições de contorno para tratamento de partículas que voam fora do limite de posição e/ou velocidade [3] para o nosso novo variante híbrido PSO.

7 Referencias

[1] G. B. Huang, Q. Y. Zhu and C. K. Siew, "Extreme Learning Machine: Theory and Applications", *Neurocomputing*, vol. 70, pp. 489-501, 2006.

[2] Q. Y. Zhu, A. K. Qin, P. N. Suganthan and G. B. Huang, "Evolutionary Extreme Learning Machine", *Pattern Recognition*, vol. 38, pp. 1759-1763, 2005.

[3] 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.

[4] 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.

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

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

[7] 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.

[8] S. Saraswathi, S. Sundaram, N. Sundararajan, M. Zimmermann and M. Nilsen-Hamilton, "ICGA-PSO-ELM Approach for Accurate Multiclass Cancer Classification Resulting in Reduced Gene Sets in Which Genes Encoding Secreted Proteins Are Highly Represented", *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 8, no. 2, pp. 452-463, 2011.

[9] M. Carvalho, T. B. Ludemir, "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.

[10] E. Eiben and J. E. Smith, "*Introduction to Evolutionary Computing*", Natural Computing Series, MIT Press, Springer, Berlin, 2003.

[11] S. Kirkpatrick, C. D. Gellat Jr. and M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, vol. 220, pp. 671-680, 1983.

[12] F. Glover, "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers and Operation Research*, vol. 13, pp. 533-549, 1986.

[13] M. Dorigo, V. Maniezzo and A. Colorni, "Ant System: Optimization by a Colony of Cooperative Agents", *IEEE Transactions on Systems, Man and Cybernetics* – Part B, vol. 26, no. 1, pp.29-41, 1996.

[14] 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.