UM MÉTODO DE PODA EFICIENTE PARA A REDE PERCEPTRON MULTICAMADAS BASEADO NA CORRELAÇÃO DOS ERROS

Cláudio M. S. Medeiros*, Guilherme A. Barreto†

*Depto. Mecatrônica, Centro Federal de Educação Tecnológica do Ceará (CEFET-Ce)
Av. 13 de Maio, 2081 - Benfica, Fortaleza, Ceará.

†Depto. Engenharia de Teleinformática, Universidade Federal do Ceará (UFC)
Av. Mister Hull, S/N - Campus do Pici, Centro de Tecnologia, Bloco 725
CP 6007, CEP. 60455-970, Fortaleza, Ceará.

Emails: claudiosa@cefetce.br, guilherme@deti.ufc.br

Abstract— In this paper we present a novel method for pruning redundant weights of a trained multilayer Perceptron (MLP). The proposed method is based on the correlation analysis of the errors produced by the output neurons and the backpropagated errors associated with the hidden neurons. Repeated applications of it leads eventually to the complete elimination of all connections of a neuron. Simulations using real-world data indicate that, in terms of performance, the proposed method compares favourably with standard pruning techniques, such as the Optimal Brain Surgeon (OBS) and Weight Decay and Elimination (WDE), but with much lower computational costs.

Keywords— MLP, model selection, regularization, pruning methods, generalization.

Resumo— Neste artigo é introduzida uma metodologia eficiente de poda de pesos desnecessários de um perceptron multicamadas (MLP) previamente treinado. O método proposto é baseado na análise da correlação entre os erros produzidos pelos neurônios de saída e os erros retropropagados associados com os neurônios da camada oculta. A aplicação sucessiva dessa metodologia de poda de pesos pode levar, eventualmente, à completa eliminação de todas as conexões de neurônios. Simulações computacionais usando dados reais indicam que o método proposto apresenta resultados equivalentes ou melhores que técnicas tradicionais de poda, como o *Optimal Brain Surgeon* (OBS) e *Weigth Decay and Elimination* (WDE), com muito menos custo computacional.

Keywords— MLP, seleção de modelos, regularização, algoritmos de poda, generalização.

1 Introdução

Muitos sistemas inteligentes para automação industrial utilizam a rede neural Perceptron multicamadas (MLP) em tarefas de predição, classificação de padrões e tomada de decisão (Dote & Ovaska 2001). Porém, ainda hoje, duas décadas após a redescoberta do algoritmo de retropropagação dos erros e apesar da existência de vasta literatura técnica sobre a rede MLP, um usuário com pouca experiência se depara com a dificuldade de estabelecer uma arquitetura ótima para uma aplicação no mundo real.

De fato, esta é uma tarefa árdua até mesmo para usuários experientes. Uma arquitetura que seja muito pequena (poucos parâmetros) pode não ser capaz de aprender apropriadamente a partir dos dados, independentemente de qual algoritmo seja usado para este propósito. Por outro lado, uma arquitetura apresentando um número demasiado de neurônios na camada oculta está sujeita a sobreajustar o modelo aos dados, modelando inclusive ruído presente nos dados.

Assim, o problema de *seleção do modelo* (Bishop 1995) é um passo crucial no projeto de uma rede MLP. Este problema, que ainda é objeto de muita atenção por parte de pesquisadores (Seghouane & Amari 2007, Curry & Morgan 2006, Nakamura et al. 2006, Xiang et al. 2005), pode ser definido como a busca pela menor arquitetura capaz de boa generalização, fazendo boas predições para dados novos. O grau de sobreajustamento (*overfitting*) e seu impacto na generalização da rede pode ser avaliado de diversas formas, inclusive através do efeito que uma mudança no número de parâmetros (pesos e limiares) causa em seu desempenho.

Dentre as várias formas de implementar essa prática, pode-se listar as três mais comuns. (*i*) *Busca exaustiva com parada prematura*: várias redes com diferentes números de neurônios na camada oculta são avaliadas em um conjunto de validação independente, a cada época de treinamento. O treinamento de cada rede é interrompido tão logo seu erro de generalização comece a crescer. A arquitetura ótima será aquela que apresenta o menor erro de generalização. O desempenho da rede selecionada deve ser confirmado em um terceiro conjunto de dados chamado de conjunto de teste. (*ii*) *Algoritmos de crescimento*: a rede começa o treinamento com um baixo número de neurônios e vai ganhando neurônios adicionais durante o processo de treinamento, objetivando atingir uma estrutura ótima. Esta é a metodologia usada, por exemplo, pela arquitetura *Cascade-Correlation* (Fahlman & Lebiere 1990). (*iii*) *Algoritmos de Poda*: a rede neural é treinada com um número relativamente grande de neurônios na camada oculta. Em seguida ocorre um processo de poda (eliminação) das conexões com pouca sig-

nificância para o desempenho da rede. A poda pode resultar até mesmo em remoção completa de neurônios. Este é o procedimento usado pelos algoritmos *Optimal Brain Surgeon* (OBS) (Fahlman & Lebiere 1990) e *Weigth Decay and Elimination* (WDE) (Bishop 1995).

Uma característica comum a todos os métodos descritos no parágrafo anterior é que eles demandam esforço computacional considerável. Por exemplo, mesmo que a busca exaustiva seja restrita a uma classe específica de arquiteturas (isto é, MLP com uma camada oculta), esta ainda é uma tarefa árdua. O algoritmo OBS, por exemplo, requer a inversão da matriz Hessiana da função de erro, tarefa que se torna bastante pesada computacionalmente, especialmente para rede com muitos parâmetros. O algoritmo WDE, por sua vez, requer a especificação, por tentativa-e-erro, de um parâmetro de regularização.

Neste artigo, introduz-se um método eficiente de poda de neurônios excedentes em uma rede MLP previamente treinada sem a necessidade de inversões de matrizes ou ajuste de quaisquer parâmetros de regularização. A motivação para o desenvolvimento desta técnica deveu-se ao fato de a rede MLP compor o módulo de detecção e classificação de falhas da plataforma MIMICO (**M***onitoramento* **I***nteligente de* **M***áquinas elétricas via* **I***nteligência* **CO***mputacional*), concebida ao longo do doutoramento do primeiro autor.

O método proposto é baseado na análise da correlação entre os erros produzidos pelos neurônios de saída e os erros retropropagados para os neurônios da camada oculta. A aplicação sucessiva dessa metodologia de poda de pesos pode levar, eventualmente, à completa eliminação de todas as conexões de alguns neurônios. Simulações computacionais usando dados reais foram realizadas para efeito de comparação com os algoritmos OBS e WDE.

O restante do artigo está organizado como segue. Na Seção 2 o algoritmo de retropropagação do erro é brevemente revisto. Na Seção 3 o método proposto é introduzido e suas principais propriedades são discuticas. Simulações são então apresentadas na Seção 4. O artigo é concluído na Seção 5 com um sumário das realizações e sugestões para trabalhos futuros.

2 O Algoritmo de Retropropagação

Esta seção traz uma breve descrição do algoritmo de retropropagação aplicado ao treinamento de um MLP completamente conectado com apenas uma camada oculta. Assim, a iteração t, a ativação do i-ésimo neurônio oculto é dada por

$$\begin{aligned} u_i^{(h)}(t) &= \sum_{j=1}^{P} w_{ij}(t)x_j(t) - \theta_i(t) \\ &= \sum_{j=0}^{P} w_{ij}(t)x_j(t), \quad i = 1, \ldots, Q \end{aligned} \tag{1}$$

na qual w_{ij} é uma conexão sináptica (peso) entre a entrada j e o neurônio i da camada oculta, $\theta_i(t)$ é o limiar do neurônio i da camada oculta, Q ($2 \leq Q < \infty$) é o número de neurônios da camada oculta e P é a dimensão do vetor de entrada (excluindo o limiar). Para simplificar a notação, faz-se $x_0(t) = -1$ e $w_{i0} = \theta_i^{(h)}(t)$.

A saída do i-ésimo neurônio oculto é então dada por

$$y_i^{(h)}(t) = \varphi_i\left[u_i^{(h)}(t)\right] = \varphi_i\left[\sum_{j=0}^{P} w_{ij}(t)x_j(t)\right], \tag{2}$$

em que $\varphi_i(\cdot)$ é uma função sigmoidal. Similarmente, as expressões para os neurônios de saída são dadas por

$$y_k^{(o)}(t) = \varphi_k\left[u_k^{(o)}(t)\right] = \varphi_k\left[\sum_{i=0}^{Q} m_{ki}(t)y_i^{(h)}(t)\right], \tag{3}$$

em que m_{ki} é o peso da conexão sináptica entre o i-ésimo neurônio oculto e o k-ésimo neurônio de saída ($k = 1, \ldots, M$), com $M \geq 1$ sendo o número de neurônios de saída. Para simplificar a notação, adota-se $y_0(t) = -1$ e $m_{k0} = \theta_k^{(o)}(t)$, em que $\theta_k^{(o)}(t)$ é o limiar do neurônio da saída k.

A etapa de retropropagação começa na camada de saída com a retropropagação dos erros $e_k^{(o)}(t) = d_k(t) - y_k^{(o)}(t)$, onde $d_k(t)$ é a saída-alvo do k-ésimo neurônio de saída. O *gradiente local* do k-ésimo neurônio de saída é dado por

$$\delta_k^{(o)}(t) = \varphi_k'\left[u_k^{(o)}(t)\right] e_k^{(o)}(t). \tag{4}$$

em que $\varphi_k'\left[u_k^{(o)}(t)\right] = \partial\varphi_k/\partial u_k^{(o)}$. Similarmente, o gradiente local $\delta_i^{(h)}(t)$ do i-ésimo neurônio oculto é calculado por

$$\begin{aligned} \delta_i^{(h)}(t) &= \varphi_i'\left[u_i^{(h)}(t)\right] \sum_{k=1}^{M} m_{ki}(t)\delta_k^{(o)}(t) \\ &= \varphi_i'\left[u_i^{(h)}(t)\right] e_i^{(h)}(t), \quad i = 0, \ldots, Q, \end{aligned} \tag{5}$$

em que o termo $e_i^{(h)}(t)$ pode ser considerado como o sinal de erro *retropropagado* ou *projetado* para o i-ésimo neurônio da camada oculta, uma vez que este são combinações lineares dos verdadeiros sinais de erro produzidos pelos neurônios de saída.

Os pesos sinápticos da rede são atualizados de acordo com as seguintes regras

$$m_{ki}(t+1) = m_{ki}(t) + \eta\delta_k^{(o)}(t)y_i^{(h)}(t), \quad (6)$$

$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_i^{(h)}(t)x_j(t), \quad (7)$$

em que $0 < \eta \ll 1$ é a taxa de aprendizagem. Uma apresentação completa de todos os dados do conjunto de treinamento durante o processo de treinamento é chamada de *época*. Muitas épocas podem ser necessárias até que haja convergência na aplicação do algoritmo de retropropagação. Assim, uma boa prática é apresentar aleatoriamente os dados do conjunto de treinamento, época a época, com o objetivo de tornar estocástica a busca no espaço de pesos durante o processo de treinamento.

Uma forma simples, porém ineficiente, de avaliar a convergência é através do erro quadrático médio (EQM)

$$\varepsilon_{train} = \frac{1}{2N}\sum_{t=1}^{N}\sum_{k=1}^{M}\left[d_k(t) - y_k^{(o)}(t)\right]^2, \quad (8)$$

calculado ao final de cada rodada de treinamento usando os próprios vetores de dados de treinamento. A convergência é obtida quando a variação do EQM se estabiliza após o decorrer de um certo número de épocas. Muitas vezes, o treinamento é interrompido taõ logo o valor do EQM caia abaixo de um valor predeterminado. O desempenho na generalização da rede MLP deve ser avaliado em um conjunto de teste, o qual deve conter amostras jamais apresentadas anteriormente à rede.

3 Metodologia Proposta

O procedimento descrito nesta seção assume que uma rede MLP é treinada para apresentar o menor valor possível de ε_{train}. Para uma rede com excesso de parâmetros, esse procedimento certamente levará a um modelo sobreajustado, com desempenho pobre na generalização. Todavia, acredita-se que o procedimento de poda eliminará conexões redundantes, melhorando assim o desempenho da rede na generalização.

3.1 Pesos entre camada oculta e de saída

O procedimento de poda começa com a submissão do conjunto de treinamento uma vez mais à rede treinada. Nenhuma adaptação de pesos ocorre neste estágio. Uma vez que todos os N exemplos de treinamento tenham sido apresentados, deve-se construir a seguintes matrizes de erro:

$$\mathbf{E}_o = \begin{bmatrix} e_1^{(o)}(1) & e_2^{(o)}(1) & \cdots & e_M^{(o)}(1) \\ e_1^{(o)}(2) & e_2^{(o)}(2) & \cdots & e_M^{(o)}(2) \\ \vdots & \vdots & \vdots & \vdots \\ e_1^{(o)}(N) & e_2^{(o)}(N) & \cdots & e_M^{(o)}(N) \end{bmatrix} \quad (9)$$

e

$$\mathbf{E}_h = \begin{bmatrix} e_0^{(h)}(1) & e_1^{(h)}(1) & \cdots & e_Q^{(h)}(1) \\ e_0^{(h)}(2) & e_1^{(h)}(2) & \cdots & e_Q^{(h)}(2) \\ \vdots & \vdots & \vdots & \vdots \\ e_0^{(h)}(N) & e_1^{(h)}(N) & \cdots & e_Q^{(h)}(N) \end{bmatrix}. \quad (10)$$

As linhas da matriz $\mathbf{E}_o$ correspondem aos erros gerados pelos neurônios de saída para um dado exemplo de treinamento. Então, esta matriz é denominada a partir de agora como *a matriz dos erros de saída*, em contraste com a matriz $\mathbf{E}_h$, cujas linhas correspondem aos erros retropropagados associados aos neurônios da camada oculta. Em particular, a primeira coluna de $\mathbf{E}_h$ corresponde aos erros retropropagados associados aos limiares $m_{k0} = \theta_k^{(o)}$, $k = 1, \ldots, M$.

O segundo passo consiste em calcular o seguinte produto de matrizes

$$\mathbf{C}_{oh} = \mathbf{E}_o^T\mathbf{E}_h, \quad (11)$$

em que o subscrito T denota o transposto da matriz. Note que o elemento (k, i) de $\mathbf{C}_{oh}$, denotado por $\mathbf{C}_{oh}[k, i]$, corresponde ao produto escalar (correlação) da k-ésima coluna de $\mathbf{E}_o$ com a i-ésima coluna de $\mathbf{E}_h$, ou seja

$$\mathbf{C}_{oh}[k, i] = \sum_{t=1}^{N} e_k^{(o)}(t)e_i^{(h)}(t), \quad (12)$$

para $k = 1, \ldots, M$, e $i = 0, \ldots, Q$.

O terceiro passo requer a ordenação dos elementos $\mathbf{C}_{oh}[k, i]$ em ordem crescente

$$\mathbf{C}_{oh}[\mathbf{r}_1] < \mathbf{C}_{oh}[\mathbf{r}_2] < \cdots < \mathbf{C}_{oh}[\mathbf{r}_L], \quad (13)$$

cujo vetor $\mathbf{r}_l = (k_l, i_l)$ contém as coordenadas do elemento de $\mathbf{C}_{oh}$ que ocupa a posição l no ranking acima, e $L = \dim(\mathbf{C}_{oh}) = M \times (Q+1)$ é o número de elementos de $\mathbf{C}_{oh}$.

O quarto passo envolve a execução do procedimento de poda mostrado na Tabela 1. Nesta tabela, J_{train} representa um certo índice de avaliação do desempenho da rede no conjunto de dados de treinamento, tal como o EQM (ε_{train}) ou a taxa de acerto de classificação (CR_{train}). A constante J_{tol} é um valor definido pelo usuário, compatível com a escolha de J_{train}.

Se $J_{train} = \varepsilon_{train}$, então J_{tol} é o máximo valor permitido do EQM para o conjunto de dados de treinamento. Assim, elimina-se uma certa conexão $m_{\mathbf{r}_l}$ apenas se o valor de J_{train}, calculado após a eliminação da mesma, permanecer abaixo de J_{tol}. No caso em que $J_{train} = CR_{train}$, então J_{tol} é a taxa de acerto mínima tolerada para o conjunto de treinamento. Nesse caso, elimina-se uma dada conexão $m_{\mathbf{r}_l}$ apenas se o valor de J_{train}, calculado após a eliminação da mesma, ainda permanecer acima de um valor previamente especificado de J_{tol}. Esta foi a escolha para a Tabela 1.

Tabela 1: Procedimento de poda para pesos entre as camadas oculta e de saída.

1.	Faça $l = 1$;	`// atribua 1 ao índice de contagem`
2.	**ENQUANTO** $l \leq L$ **FAÇA**	`// comece o ciclo de poda`
	2.1. Faça $a = m_{\mathbf{r}_l}$;	`// salve o valor corrente`
	2.2. Faça $m_{\mathbf{r}_l} = 0$;	`// atribua zero ao peso que se quer eliminar`
	2.3. Calcule J_{train};	`// índice de desempenho no conjunto de treinamento`
	2.4. **SE** $J_{train} < J_{tol}$,	`// Adotou-se` $J_{train} = CR_{train}$
	ENTÃO Faça $m_{\mathbf{r}_l} = a$;	`// recupere o valor do peso`
	Faça $l = L + 1$;	`// interrompa a poda`
	2.5. Faça $l = l + 1$;	`// continue a poda`

3.2 Pesos entre a entrada e camada oculta

Para a poda dos pesos que conectam as unidades de entrada aos neurônios da camada oculta, w_{ij}, é necessário retropropagar os erros relacionados aos neurônios ocultos, $e^{(h)}(t)$, a fim de se obter os erros projetados nas unidades de entrada, ou seja

$$e_j^{(i)}(t) = \sum_{i=1}^{Q} w_{ij}(t)\delta_i^{(h)}(t), \quad j = 0, \ldots, P. \quad (14)$$

Após a apresentação dos N vetores de treinamento, os erros resultantes podem ser organizados numa matriz de erros

$$\mathbf{E}_i = \begin{bmatrix} e_0^{(i)}(1) & e_1^{(i)}(1) & \cdots & e_P^{(i)}(1) \\ e_0^{(i)}(2) & e_1^{(i)}(2) & \cdots & e_P^{(i)}(2) \\ \vdots & \vdots & \vdots & \vdots \\ e_0^{(i)}(N) & e_1^{(i)}(N) & \cdots & e_P^{(i)}(N) \end{bmatrix}. \quad (15)$$

De modo semelhante ao procedimento descrito na seção anterior, faz-se necessário calcular o seguinte produto de matrizes

$$\mathbf{C}_{hi} = \mathbf{E}_h^T \mathbf{E}_i, \quad (16)$$

no qual o (i, j)-ésimo elemento de $\mathbf{C}_{hi}$, denotado por $\mathbf{C}_{hi}[i, j]$, é dado por

$$\mathbf{C}_{hi}[i, j] = \sum_{t=1}^{N} e_i^{(h)}(t) e_j^{(i)}(t), \quad (17)$$

para $i = 1, \ldots, Q$, and $j = 0, \ldots, P$. Então, os elementos $\mathbf{C}_{hi}[i, j]$ são ordenados em ordem crescente

$$\mathbf{C}_{hi}[\mathbf{s}_1] < \mathbf{C}_{hi}[\mathbf{s}_2] < \cdots < \mathbf{C}_{hi}[\mathbf{s}_L], \quad (18)$$

tal que o vetor $\mathbf{s}_l = (i_l, j_l)$ contém as coordenadas do elemento de $\mathbf{C}_{hi}$ que ocupa a posição l no ranking acima, e $L = \dim(\mathbf{C}_{hi}) = M \times (P + 1)$ é o número de elementos em $\mathbf{C}_{hi}$.

O passo final envolve a execução do procedimento de poda mostrado na Tabela 2. Para esta tabela, nós também usamos $J_{train} = CR_{train}$. Por razões óbvias, a partir de agora o método proposto será chamado de CAPE (**C**orrelation **A**nalysis of back-**P**ropagated **E**rrors).

4 Simulações e Discussões

Para avaliar o método CAPE é utilizado um conjunto de dados biomédicos gentilmente disponibilizados para esta pesquisa pelo *Grupo de Pesquisa Aplicada em Ortopedia* (GARO) do *Centro Médico-Cirúrgico de Readaptação de Massues*, Lyon, França. Estes dados também foram utilizados por Neto et al. (2006).

O problema consiste em diagnosticar pacientes como pertencentes a uma de três categorias: Normal (100 pacientes), Hénia de Disco (60 pacientes) ou Espondilolistese (150 pacientes). Cada paciente é representado na base de dados por seis atributos biomecânicos relacionados com a forma e orientação da pélvis e da coluna vertebral: incidência pélvica, ângulo de versão pélvica, declive sacral, raio pélvico, ângulo de lordose lombar e grau de espondilolistese. Os seis atributos numéricos foram medidas por um médico ortopedista a partir de imagens de raio-X da coluna vertebral. Maiores detalhes sobre esses atributos bem como sua relação com as patologias na coluna vertebral podem ser encontrados em Berthonnaud et al. (2005).

Um total de 42 padrões de vetores foram selecionados aleatoriamente de cada classe para constituir o conjunto de treinamento. Os exemplos restantes (184) foram usados para fins de teste. As dimensões da entrada e saída foram definidas em $P = 6$ e $M = 3$, respectivamente. Pesos e limiares foram iniciados aleatoriamente dentro da faixa $[-0,5 \text{ - } 0,5]$. Todos os neurônios usaram função de ativação hiperbólica e as entradas foram normalizadas entre -1 e $+1$. Os vetores com as saídas desejadas (rótulos) para as três classes foram definidos como: $\mathbf{d} = [0,98\ -0,98\ -0,98]^T$ para a classe Normal, $\mathbf{d} = [-0,98\ 0,98\ -0,98]^T$ para Hérnia de Disco e $\mathbf{d} = [-0,98\ -0,98\ 0,98]^T$ para Espondilolistese. A taxa de aprendizagem foi especificada como $\eta = 0,001$. A rede foi treinada até que o EQM para o conjunto de treinamento (ε_{train}) estabilizasse no menor valor possível para uma dada rodada de treinamento.

Para esta simulação, adotou-se $PI_{tol} = CR_{tol} = 85\%$. Este valor serve como referência para a taxa de reconhecimento mínima aceitável para os conjuntos de treinamento e teste. Isto é,

Tabela 2: Procedimento de poda dos pesos entre as unidades de entrada e a camada oculta.

1. Faça $l = 1$;	// atribua 1 ao índice de contagem
2. **ENQUANTO** $l \leq L$ **FAÇA**	// comece o ciclo de poda
2.1. Faça $a = m_{\mathbf{r}_l}$;	// salve o valor corrente
2.2. Faça $m_{\mathbf{r}_l} = 0$;	// atribua zero ao peso
2.3. Calcule J_{train};	// índice de desempenho no conjunto de treinamento
2.4. **SE** $J_{train} < J_{tol}$,	// Adotou-se $J_{train} = CR_{train}$
ENTÃO Faça $m_{\mathbf{r}_l} = a$;	// recupere o valor
Faça $l = L + 1$;	// interrompa a poda
2.5. Faça $l = l + 1$;	// continue a poda

Tabela 3: Resultados numéricos da aplicação dos métodos de poda.

	Q	N_c	CR_{train}	CR_{test}	ε_{train}	ε_{test}	AIC
Arquitetura 1	24	243	89.68	87.50	0.1132	0.1274	490.36
CAPE	19	126	92.06	86.96	0.1662	0.1273	255.59
OBS	21	133	92.06	84.78	0.1837	0.2132	265.39
PWM	21	121	89.68	84.24	0.1733	0.1544	245.51
WDE	24	219	78.57	80.98	0.4237	0.4176	439.72
Arquitetura 2	18	183	96.03	86.41	0.0616	0.1891	371.57
CAPE	14	98	95.24	88.59	0.1089	0.1632	200.43
OBS	16	103	95.24	82.07	0.1184	0.2014	210.26
PWM	16	96	96.03	88.59	0.1453	0.1918	195.86
WDE	16	158	85.71	90.76	0.1747	0.1132	319.49
Arquitetura 3	13	133	93.65	81.52	0.0765	0.2311	271.14
CAPE	13	103	93.65	83.70	0.1211	0.2113	210.22
OBS	13	102	93.65	80.43	0.1267	0.2410	208.13
PWM	13	96	93.65	78.80	0.1356	0.2348	196.00
WDE	12	108	65.08	69.02	0.4323	0.4292	217.68

mesmo que a rede podada produza um valor de CR_{train} maior que CR_{tol}, seu valor CR_{test} também deve ser maior que CR_{tol}.

A poda da rede é realizada através da aplicação sucessiva do método CAPE. Resultados numéricos são mostrados na Tabela 3. Nesta tabela, N_c é o número de conexões existentes, CR_{train} e CR_{test} representam as taxas de acerto na classificação dos dados de treinamento e teste, respectivamente, e AIC é o valor do critério de informação de Akaike[1] para uma dada arquitetura. O acrônimo PWM significa *Pruning by Weight Magnitude*, o qual é um método de poda baseado na eliminação dos pesos cujas magnitudes sejam muito pequenas em relação à magnitude dos demais pesos (Bishop 1995). O procedimento adotado para PWM é similar ao adotado pelo método CAPE. Primeiramente, os pesos são ordenados em ordem crescente de seus valores absolutos. Então, partindo do menor, executa-se a poda dos pesos enquanto CR_{train} não for inferior a CR_{tol}.

Inicialmente, define-se como $Q = 24$ o número de neurônios ocultos de uma rede MLP completamente conectada com apenas uma camada oculta. Esta rede, denominada Arquitetura 1, tem $N_c = 243$ parâmetros ajustáveis ao todo. Observe que esta arquitetura inicial, uma vez treinada, classifica os dados precisamente. Entretando, não se está interessado apenas em boas taxas de classificação, mas também em saber se a arquitetura está sobredimensionada.

O algoritmo CAPE podou a Arquitetura 1 de tal forma que reduziu a camada oculta a $Q = 19$ neurônios e a $N_c = 126$ conexões. Portanto, a rede resultante apresenta o número de conexões inferior a uma rede com $Q = 19$ completamente conectada (i.e. $N_c = 193$). A aplicação do método OBS resultou numa rede podada com $Q = 21$ neurônios na camada oculta e $N_c = 133$. A taxa de acerto de ambos os métodos no treinamento foi a mesma, porém no teste apenas a rede podada pelo método CAPE manteve sua taxa de acerto acima de 85%.

Em relação ao método PWM, deve-se mencionar que, independentemente do fato de ele ter obtido uma rede com um número menor de conexões ($N_c = 121$) para $Q = 21$ neurônios na camada oculta, seus valores para CR_{train} e CR_{test} foram os piores em comparação com o CAPE e o OBS. Além disso, devido ao baixo número de conexões ele atingiu o mais baixo valor do AIC, indicando errôneamente que esta arquitetura é a melhor para este problema. Esses resultados sugerem que o índice AIC pode não ser a melhor indicação para seleção de modelos em problemas de

[1] $AIC = -2\ln(\varepsilon_{train}) + 2N_c$ (Principe et al. 2000).

classificação, pois o mesmo leva em consideração os valores do EQM e não as taxas de classificação.

Considerando que o desempenho da rede resultante da aplicação do método CAPE à Arquitetura 1 permanece dentro de níveis aceitáveis, divagou-se acerca do que poderia acontecer se uma outra rede completamente conectada (Arquitetura 2), com o número de neurônios ocultos próximo ao obtido pela poda da Arquitetura 1, fosse submetida ao processo de poda. A idéia é verificar se ainda existe espaço para poda. Os parâmetros de treinamento da Arquitetura 2 foram os mesmos anteriormente utilizados, com pesos e limiares iniciados aleatoriamente.

A aplicação do método de poda CAPE à Arquitetura 2 resulta no menor número de neurônios ocultos ($Q = 14$), com menos conexões que a rede original. Em comparação ao OBS, também apresenta maior taxa de classificação CR_{test} e melhor índice AIC. O PWM apresentou bons resultados relativamente a taxas de classificação, finalizando com uma arquitetura podada com $Q = 16$ neurônios na camada oculta e $N_c = 96$ conexões. É importante mencionar, entretanto, que o PWM é bastante sensível à inicialização dos pesos, e que os resultados numéricos mostrados na Tabela 3 correspondem ao melhor dentre 10 tentativas de treinamento. Os métodos CAPE e OBS são muito menos sensíveis às condições iniciais dos pesos.

Como a taxa de acerto CR_{test} da rede resultante da aplicação da poda à Arquitetura 2 ainda permanece acima de $CR_{tol} = 85\%$, decidiu-se por reiniciar, treinar e podar uma nova rede (Arquitetura 3), totalmente conectada com $Q = 13$ neurônios ocultos. É importante salientar que nenhum método foi capaz de reduzir o número de neurônios na camada oculta, mas apenas o número de conexões. Isto pode ser um indicativo de que a rede já está bem "enxuta" em termos de neurônios ocultos, dado espaço apenas para um ajuste fino em termos de eliminação de umas poucas conexões redundantes. De qualquer forma, todos os métodos de poda produziram taxas de reconhecimento CR_{train} e CR_{test} abaixo do valor considerado como aceitável. Assim, pode-se inferir que as arquiteturas mais adequadas para o problema em questão são aquelas resultantes da poda da Arquitetura 2.

Os melhores resultados obtidos para a aplicação do algoritmo WDE para as Arquiteturas 1, 2 e 3 são também mostradas na Tabela 3. Foi adotado $\lambda = 0,001$ após alguma experimentação. O desempenho do algoritmo WDE foi muito pobre para todas as arquiteturas. Para fins de comparação, foi implementado o método de *Busca exaustiva com parada prematura* numa rede (MLP) com apenas uma camada oculta totalmente conectada. Após várias tentativas, atingiu-se uma rede com $Q = 12$ ($N_c = 123$) e $CR_{test} = 83,57$. A despeito do fato de que o método conduziu a uma rede com um menor número de neurônios na camada oculta do que o método CAPE ($Q = 14$ e $N_c = 98$), este último obteve taxa de reconhecimento equivalente com menos conexões.

5 Conclusões

Este artigo introduziu o método CAPE, um procedimento eficiente e de fácil aplicação para poda de pesos desnecessários de uma rede MLP previamente treinada. O método CAPE baseia-se na análise de correlação entre os erros produzidos nos neurônios de saída e os erros retropropagados associados aos neurônios da camada oculta.

Simulações usando dados reais indicaram que, em termos de desempenho, o método CAPE tem desempenho comparável ou melhor que técnicas de poda convencionais, tais como OBS, WDE e parada antecipada com validação cruzada, possuindo também menor esforço computacional. O método de poda baseado na magnitude dos pesos também foi avaliado. Este, apesar de algumas vezes apresentar bons resultados, é muito sensível aos valores iniciais dos pesos e limiares.

Agradecimentos: Os autores agradecem à CAPES/PRODOC pelo apoio financeiro.

Referências

Berthonnaud, E., Dimnet, J., Roussouly, P. & Labelle, H. (2005). Analysis of the sagittal balance of the spine and pelvis using shape and orientation parameters, *Journal of Spinal Disorders & Techniques* **18**(1): 40–47.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*, Oxford University Press.

Curry, B. & Morgan, P. H. (2006). Model selection in neural networks: Some dificulties, *European Journal of Operational Research* **170**(2): 567–577.

Dote, Y. & Ovaska, S. J. (2001). Industrial applications of soft computing: A review, *Proceedings of the IEEE* **89**(9): 1243–1265.

Fahlman, S. E. & Lebiere, C. (1990). The cascade-correlation learning architecture, *in* D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems*, Vol. 2, Morgan Kaufmann, San Mateo, pp. 524–532.

Nakamura, T., Judd, K., Mees, A. I. & Small, M. (2006). A comparative study of information criteria for model selection, *International Journal of Bifurcation and Chaos* **16**(8): 2153–2175.

Neto, A. R. R., Barreto, G. A., Cortez, P. C. & Mota, H. (2006). SINPATCO: Sistema inteligente de diagnóstico de patologias da coluna vertebral, *Anais do XVI Congresso Brasileiro de Automática (CBA'06)*, Salvador, BA, pp. 929–934.

Principe, J. C., Euliano, N. R. & Lefebvre, W. C. (2000). *Neural and Adaptive Systems*, John Wiley & Sons.

Seghouane, A.-K. & Amari, S.-I. (2007). The AIC criterion and symmetrizing the Kullback-Leibler divergence, *IEEE Transactions on Neural Networks* **18**(1): 97–106.

Xiang, C., Ding, S. Q. & Lee, T. H. (2005). Geometric interpretation and architecture selection of the MLP, *IEEE Transactions on Neural Networks* **16**(1): 84–96.