Espaço de Pesos com Geometria Riemanniana para Aceleração da Convergência de Perceptron Multicamada

Luciano Frontino de Medeiros e Hamilton Pereira da Silva, *UNINTER-PR*

***Abstract*—The convergence of a multilayer perceptron toward a global minimum error in the backpropagation algorithm can be accelerated by introducing the representation of weight search space in accordance with the Riemann geometry, which consider curved spaces. The trajectory of the search vector at the training phase is modified by a "force" originated from the curvature of weight space. The main idea is inspired at the General Relativity theory, and the search space (now a manifold) must be controlled by parameters like spatial density and scale factor, modifying the gradient descent in the backpropagation algorithm.**

***Index Terms*—Neural Networks, Riemann Geometry, Curved Spaces, Tensorial Calculus, General Relativity, Backpropagation.**

I. Introdução

NO decorrer do desenvolvimento da área de redes neurais, vários modelos físicos foram aplicados a arquiteturas ou algoritmos de treinamento. A convergência de redes neurais, por sua vez, é também um campo bastante explorado. Com relação ao perceptron multicamada, foram desenvolvidas diversas técnicas que proporcionam rapidez e performance no treinamento em contrapartida ao algoritmo clássico da retropropagação por erro[2][3][4]. Algumas técnicas desenvolvidas incluem algoritmos de segunda ordem, método do gradiente conjugado e algoritmo Levenberg-Marquardt [5][6][17][18][19]. Pode-se considerar ainda os métodos de informação prévia de gradiente [7][8], de passo adaptativo [9][10][11], inicialização apropriada de pesos[12][13][14], processos de otimização baseada em mínimos quadrados[14][15], e ainda métodos híbridos[16].

A representação geométrica da superfície de erro em espaço de pesos para auxiliar a visualização da trajetória do aprendizado também é objeto de estudo da área de redes neurais [20][21][22]. Neste trabalho, não se propõe um novo algoritmo mas sim a substituição do espaço de busca de pesos, que é considerado como um espaço plano ou *cartesiano*, por um espaço *riemanniano*, o qual é descrito por uma métrica que considera espaços curvos. A abordagem utilizou conceitos provenientes da teoria da Relatividade Geral (GR), a qual postula que o espaço-tempo quadridimensional se curva na presença de matéria-energia [24][25][26][27]. Explorou-se, portanto, a característica de curvatura de um espaço com geometria riemanniana com o objetivo de acelerar o processo de convergência para o erro mínimo global, sem interferir na base do algoritmo de aprendizado por retropropagação. A abordagem aqui estará restrita ao perceptron multicamada, podendo ser estendida a outros tipos de redes.

Para a obtenção das expressões do algoritmo da descida de gradiente modificada, foram utilizados conceitos gerais do Cálculo Tensorial[27][28][29]. A abordagem matemática adotada foi iniciar por uma definição prévia de métrica de acordo com certas características desejadas para o comportamento do espaço e, a partir daí, deduzir-se a geometria do espaço para uma curvatura fechada, trabalhando-se com uma solução não homogênea para as equações de curvatura.

O trabalho será exposto nas seguintes fases: definição do espaço de busca, definição de regras para o modelo de espaço riemanniano, desenvolvimento de uma solução métrica, transformação entre os espaços riemanniano e plano e descida de gradiente modificada. Ao final se apresenta uma seção com aplicação prática mostrando os resultados preliminares obtidos e a conclusão.

II. O Espaço de Busca de Pesos

O processo que será abordado aqui está de acordo com o formalismo descrito em [1]. O erro médio quadrático global de um perceptron multicamada (que será simplificado no decorrer do texto como MLP, das iniciais do inglês *Multilayer Perceptron*) resulta do somatório dos erros quadráticos parciais através da apresentação das amostras de treinamento às camadas da rede. Sendo E o erro global, e o erro parcial, d o valor desejado e y o valor atual, a relação entre elas pode ser expressa por

$$E^2 = \sum_j e_j^2 = \sum_j (d_j - y_j)^2 \tag{1}$$

A obtenção de y faz-se mediante a função de ativação f da combinação linear u_j,

$$y_j = f(u_j) \tag{2}$$

Sendo u_j calculado como

$$u_j = \sum_{j,i} w_{ij} x_i \tag{3}$$

com x sendo os valores dos neurônios de entrada da rede, e w_{ij} sendo os pesos da mesma. Uma visualização geométrica do processo de treinamento pode ser construída a partir da representação em eixos de um sistema de coordenadas dos pesos w_{ij} e do erro médio quadrático global E. O processo dinâmico que surge desta abstração, a cada passo de treinamento, é o de um vetor percorrendo uma (hiper)superfície em busca de um ponto de mínimo. Este sistema de coordenadas é considerado como o *espaço de busca de pesos*. Uma representação de um vetor V neste espaço pode ser $V(E, w_{11}, w_{12}, ..., w_{ij}, ..., w_{MP})$, $i \in [1, M]$ e $j \in [1, P]$. Este espaço possui o número de dimensões (ou cardinalidade) de valor $MP + 1$, com $w_{ij} \in \mathbb{R}$ e

Luciano Frontino de Medeiros atualmente é professor e coordenador de curso no grupo UNINTER-PR, e-mail: luciano@facinter.br

Hamilton Pereira da Silva atualmente é professor e diretor de tecnologia no grupo UNINTER-PR, e-mail: hsilva@facinter.br

$E \in \mathbb{R}^*_+$. Na maior parte do formalismo apresentado adiante para a obtenção da solução métrica, o índice duplo será substituído por um único índice para fins de simplificação.

De acordo com o algoritmo de retropropagação [3][4], durante o processo de treinamento de um MLP acontecem modificações sucessivas em direção a um ponto de mínimo. Isto pode ser expresso pelas seguintes fórmulas

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)} \tag{4}$$

e

$$\Delta w_{ij}^{(t)} = \alpha \Delta w_{ij}^{(t-1)} - \eta \frac{\partial E}{\partial w_{ij}}^{(t-1)} \tag{5}$$

onde η refere-se à taxa de aprendizado, α refere-se à taxa de momento, e $\partial E/\partial w_{ij}$ o gradiente do erro em relação ao peso w_{ij} [4], considerando t o número de épocas[1][23]. Portanto, as equações (4) e (5) serão o alvo para as modificações a serem propostas conforme o desenvolvimento explicado a seguir.

III. Um Modelo de Espaço de Pesos riemanniano

Para a definição de uma métrica, faz-se necessário a definição de certas "regras" para o comportamento do espaço. Assim, duas regras serão adotadas para nortear a definição de um modelo de espaço riemanniano e por conseqüência a sua respectiva métrica:

1. A curvatura do espaço de pesos aumenta quanto mais distante o vetor de busca estiver do ponto de mínimo. Esta afirmação equivale à suposição do ponto de mínimo como sendo o centro de um atrator para o vetor de busca, com a "força" que atua sobre o vetor surgindo a partir da curvatura do espaço;
2. Quanto mais próximo da fase final de treinamento, a curvatura do espaço gradativamente desaparece, retornando o vetor de busca ao espaço de pesos plano.

Para a obtenção do espaço riemanniano, é necessária a expressão da métrica do espaço na seguinte forma quadrática:

$$ds^2 = g_{ij} dx^i dx^j \tag{6}$$

onde dx^i são os deslocamentos infinitesimais nas coordenadas x^i, e g_{ij} é o tensor métrico do espaço. Para um espaço plano normal, tem-se $g_{ii} = 1$ e $g_{ij} = 0, i \neq j$. Os índices sobrescritos referem-se a quantidades *contravariantes* e os subscritos referem-se a quantidades *covariantes*, de acordo com o Cálculo Tensorial.

A seguir, calculam-se os *símbolos de Christoffel* ou *conexões métricas* a partir da fórmula:

$$\Gamma^i_{jk} = \frac{1}{2} g^{im} \left(\frac{\partial g_{km}}{\partial x^j} + \frac{\partial g_{mj}}{\partial x^k} - \frac{\partial g_{jk}}{\partial x^m}\right) \tag{7}$$

A partir das conexões métricas calcula-se o *tensor de curvatura* ou *tensor de Riemann-Christoffel*:

$$R^\rho{}_{ijk} = \frac{\partial \Gamma^\rho_{ki}}{\partial x^j} - \frac{\partial \Gamma^\rho_{ji}}{\partial x^k} + \Gamma^\rho_{jm}\Gamma^m_{ki} - \Gamma^\rho_{km}\Gamma^m_{ji} \tag{8}$$

Pela operação de contração de tensores, obtém-se o *tensor de Ricci* (e trocando-se k por j):

$$R_{ij} = R^\rho{}_{i\rho k} \tag{9}$$

Para que a métrica seja *fechada*[1], a derivada covariante do tensor de curvatura deve ser igual a zero. Com o objetivo de simplificar o desenvolvimento e sem abordar o conceito de derivada covariante, trabalharemos com o tensor de Ricci requerendo a condição $R_{ij} = 0$ (condição que é tratada na GR como as equações de campo no vácuo). Isto estará de acordo com a teoria e será suficiente agora para calcularmos a solução a ser trabalhada na seqüência.

IV. Obtenção do Tensor Métrico

Define-se o espaço de busca de pesos S com o sistema de coordenadas $(E, w_1, ..., w_n)$, $w_i \in \mathbb{R}$ e $E \in \mathbb{R}^*_+$, com $n+1$ dimensões ortogonais (por enquanto é feito o desenvolvimento com apenas um único índice subscrito). Assim como enunciado em (6), o quadrado do elemento de linha deste espaço é dado por

$$ds^2 = dE^2 + \sum_{i=1}^{n} dw_i^2 \tag{10}$$

Será preciso desenvolver uma métrica genérica na forma

$$ds^2 = e^{2\alpha} dE^2 + \sum_{i=1}^{n} e^{2\beta}(dw_i^2) \tag{11}$$

Esta abordagem é baseada na existente na GR para a obtenção da métrica de Schwarzchild [24][25]. As funções α e β variam em função da coordenada E. De posse da métrica, define-se o tensor métrico. A função exponencial em α estaria relacionada à coordenada do erro global E, e a função exponencial em β seria comum às coordenadas dos pesos (O fator 2 no expoente consta para evitar o uso de frações). Por enquanto, limita-se a $n = 2$, de forma que

$$ds^2 = e^{2\alpha} dE^2 + e^{2\beta}(dw_1^2 + dw_2^2) \tag{12}$$

Com a métrica diferenciada, os componentes do tensor métrico não nulos são,

$$g_{EE} = e^{2\alpha}, g_{ii} = e^{2\beta} \tag{13}$$

Com isto, pode-se agora calcular as conexões métricas. Com base na equação (7), obtemos as seguintes conexões métricas diferentes de zero

$$\Gamma^E_{EE} = \alpha' \tag{14}$$

$$\Gamma^E_{11} = \Gamma^E_{22} = -\beta' e^{2(\beta-\alpha)} \tag{15}$$

$$\Gamma^1_{E1} = \Gamma^1_{1E} = \Gamma^2_{E2} = \Gamma^2_{2E} = \beta' \tag{16}$$

O apóstrofe indica a derivada em relação à variável E. As conexões métricas relacionadas às coordenadas w_1 e w_2 são igual a zero, pois as funções α e β dependem apenas de E.

Na seqüência, verificam-se os componentes do tensor de curvatura ou Riemann-Christoffel. Dois componentes do tensor de curvatura não nulos são

$$R^E{}_{1E1} = R^E{}_{2E2} = -e^{2(\beta-\alpha)}(\beta'' + \alpha'\beta' - \beta'^2) \tag{17}$$

Através das operações de contração e rebaixamento de índices apropriadas, chega-se na expressão do tensor de Ricci, onde os

[1] Ou seja, para que a base de coordenadas possa ser aplicada de maneira uniforme a todo o espaço.

demais componentes R_{ii} do tensor de Ricci podem ser expressos em função do componente R_{EE}:

$$R_{EE} = g^{11}R_{E1E1} = -(\beta'' + \alpha'\beta' - \beta'^2) \tag{18}$$

$$R_{11} = R_{22} = R_{EE}e^{2(\beta-\alpha)} \tag{19}$$

Percebe-se assim que a quantidade linearmente independente a considerar no cálculo do tensor de Ricci é o componente R_{EE}. Para um espaço riemanniano com curvatura fechada, conforme exposto anteriormente, requer-se a condição $R_{EE} = 0$. Desta forma e, de acordo com (18), a resolução da equação diferencial homogênea não linear irá resultar em expressões de α e β necessárias para a definição da métrica:

$$\frac{d^2\beta}{dE^2} + \frac{d\alpha}{dE}\frac{d\beta}{dE} - (\frac{d\beta}{dE})^2 = 0 \tag{20}$$

É interessante notar que não importa o número de coordenadas de pesos w_i; a equação acima *mantém-se inalterada*, o que será determinante para inferir o algoritmo de retropropagação para um MLP com uma quantidade maior de pesos. Baseando-se na equação (20), faz-se $\alpha = 0$, indicando que a coordenada do erro global E estará livre da influência da curvatura. Portanto, de posse da equação diferencial homogênea

$$\frac{d^2\beta}{dE^2} - (\frac{d\beta}{dE})^2 = 0 \tag{21}$$

A solução obtida a seguir para β introduz os parâmetros κ e ρ no lugar das constantes de integração,

$$\beta = +\ln(1 + \kappa\rho E) \tag{22}$$

E os termos da métrica riemanniana para o espaço de busca de pesos ficam:

$$g_{EE} = 1, g_{ii} = (1 + \kappa\rho E)^2 \tag{23}$$

Em termos práticos, a seguinte solução não homogênea atende melhor a proposta do modelo riemanniano desejado:

$$\beta = -\ln(1 + \kappa\rho E) \tag{24}$$

e por conseqüência os termos da métrica ficam:

$$g_{EE} = 1, g_{ii} = \frac{1}{(1 + \kappa\rho E)^2} \tag{25}$$

E desde que o tensor de Ricci R_{ij} venha a ser proporcional a um dado tensor T_{ij}, com a seguinte relação entre tais componentes

$$R_{EE} = kT_{EE} \tag{26}$$

com

$$T_{EE} = e^{2\beta}, \tag{27}$$

tal solução para β introduzirá variações nas coordenadas dos pesos proporcionais à coordenada E, de acordo com as regras expostas na seção anterior. Considera-se ρ um parâmetro que terá um comportamento semelhante à "densidade de matéria-energia" da Relatividade Geral, que denominaremos aqui de *densidade do espaço* ou *densidade espacial*, e κ o *fator de escala* para E. A razão para isto é que κ possuiria um valor que ponderaria o erro global E de acordo com o valor inicial dos pesos no algoritmo de treinamento. A densidade de espaço ρ sofreria variações de acordo com a execução do treinamento, sendo modificado dentro do próprio algoritmo de retropropagação. Deve-se concluir que esta solução não homogênea para β apresenta as seguintes propriedades com relação à coordenada E:

1. *Confinamento* - à medida em que o valor do erro global aumenta, β tende a ser infinitamente pequeno, e o termo da métrica se anula, ficando o vetor "preso" no espaço riemanniano.

$$\lim_{E\to+\infty} g_{ii} = 0 \tag{28}$$

2. *Resiliência* - próximo do valor de erro global mínimo, os espaços riemanniano e plano tendem a ser iguais.

$$\lim_{E\to 0} \beta = 0 \;\; e \;\; \lim_{E\to 0} g_{ii} = 1 \tag{29}$$

As propriedades de confinamento e resiliência desta solução para β atendem respectivamente às regras 1 e 2 enunciadas na seção III.

V. Transformação de Coordenadas e Expressão do Gradiente

A transformação de coordenadas é necessária para convertermos as expressões de atualização dos pesos (4) e (5) para o espaço riemanniano. Ainda, um mapeamento perfeito entre os espaços deve ser suave, difeomórfico e inversível[24]. Para se expressar o mapeamento ϕ para os espaços em questão, parte-se para a equivalência entre as métricas. Para o espaço S^n adota-se a a expressão da métrica (23) do espaço riemanniano:

$$ds^2 = dE^2 + \frac{1}{(1 + \kappa\rho E)^2}dw_{ij}^2 \tag{30}$$

Para o espaço $S^{'n}$, considera-se a métrica do espaço plano

$$ds^{'2} = dE^{'2} + dw_{ij}^{'2} \tag{31}$$

Para generalizar o desenvolvimento, substitui-se a variável de um índice nos pesos por w_{ij}. Comparando as métricas (30) e (31), a equivalência entre os dois espaços estará associada pela transformação $\phi : S^n \to S^{'n}$, embasada nas propriedades na seção III, dada por

$$\begin{cases} E' = \sqrt{g_{EE}}E = E \\ w_{ij}^{'} = \sqrt{g_{kk}}w_{ij} = \frac{w_{ij}}{(1+\kappa\rho E)}, k = 1, .., MP. \end{cases} \tag{32}$$

Para expressar o algoritmo de retropropagação modificado, ainda é preciso representar o gradiente no espaço de pesos riemanniano. De acordo com o Cálculo Tensorial, se φ é definido como um escalar (um tensor de posto zero), suas derivadas parciais irão formar um novo tensor (ou vetor) covariante. Este tensor é definido como o *gradiente* do escalar φ [26],[27], [28] [29]. Assim

$$\frac{\partial\varphi}{\partial u_j^{'}} = \frac{\partial\varphi}{\partial u_k}\frac{\partial u_k}{\partial u_j^{'}} = (\nabla\varphi)_j \tag{33}$$

Considerando a transformação dada por $\phi : S(E, w_1...w_n) \rightarrow S'(E', w_1^{'}...w_n^{'})$, e substituindo de acordo com (32), tem-se

$$\frac{\partial \varphi}{\partial E'} = \frac{\partial \varphi}{\partial E}\frac{\partial E}{\partial E'} = \frac{1}{\sqrt{g_{EE}}}\frac{\partial \varphi}{\partial E} \tag{34}$$

$$\frac{\partial \varphi}{\partial w_i^{'}} = \frac{\partial \varphi}{\partial w_i}\frac{\partial w_i}{\partial w_i^{'}} = \frac{1}{\sqrt{g_{ii}}}\frac{\partial \varphi}{\partial w_i} \tag{35}$$

Fazendo com que o campo escalar φ seja coincidente com a coordenada E, obtemos:

$$\frac{\partial E}{\partial E'} = \frac{1}{\sqrt{g_{EE}}} = 1 \tag{36}$$

$$\frac{\partial E}{\partial w_i^{'}} = \frac{1}{\sqrt{g_{ii}}}\frac{\partial E}{\partial w_i} = (1 + \kappa\rho E)\frac{\partial E}{\partial w_i} \tag{37}$$

As expressões (36) e (37) mostram a equivalência entre os vetores gradiente nos espaços S e S', sendo que (37) interessa para o procedimento de atualização dos pesos pela descida de gradiente.

VI. Descida de Gradiente no Espaço Riemanniano

A expressão para a atualização dos pesos no espaço S' é a expressão genérica aplicada para o algoritmo de retropropagação [1][23], considerando t a época atual de treinamento e $t+1$ a época sucessiva,

$$w_i^{'(t+1)} = w_i^{'(t)} - \eta\frac{\partial E}{\partial w_i^{'}} \tag{38}$$

Para a equivalência no espaço S, substituímos conforme a regra de transformação definida em (32) e de acordo com (37),

$$\sqrt{g_{ii}}w_i^{(t+1)} = \sqrt{g_{ii}}w_i^{(t)} - \eta\frac{1}{\sqrt{g_{ii}}}\frac{\partial E}{\partial w_i} \tag{39}$$

Ou

$$w_i^{(t+1)} = w_i^{(t)} - \eta\frac{1}{g_{ii}}\frac{\partial E}{\partial w_i} \tag{40}$$

Utilizando a métrica definida em (23), obtemos

$$w_1^{(t+1)} = w_i^{(t)} - \eta(1 + \kappa\rho E)^2\frac{\partial E}{\partial w_i} \tag{41}$$

É possível fazer tal como em (5) a representação da atualização dos pesos mais completa com o termo de inércia α:

$$\Delta w_{ij}^{(t)} = \alpha\Delta w_{ij}^{(t-1)} - \eta(1 + \kappa\rho E)^2\frac{\partial E}{\partial w_{ij}}^{(t-1)} \tag{42}$$

Tal expressão substitui a atualização clássica dos pesos, sendo necessário definir o padrão de comportamento da densidade de espaço ρ no treinamento. Deve ser ressaltado que a presença do termo de correção em ρ e E induz a uma "aceleração" do vetor de busca. Portanto, o valor da taxa de aprendizado η deve ser sensivelmente alterado para evitar instabilidades no processo de treinamento.

Para adequar a representação para redes MLP com camadas ocultas, o segundo termo de (42) pode ser expresso como [1]

$$\Delta w_{ij}^{(t)} = \alpha\Delta w_{ij}^{(t-1)} - \eta\delta_i^{(t-1)}x_j^{(t-1)} \tag{43}$$

onde o termo de *gradiente local* δ_i (suprimindo-se o índice n referente à época de treinamento) do neurônio i é dado em função do erro local e_i e da função de ativação f,

$$\delta_i = (1 + \kappa\rho E)^2 e_i f_i^{'}(u_i). \tag{44}$$

Esta expressão deve ser utilizada na camada de saída da rede, sendo que nas camadas ocultas têm-se o gradiente local do neurônio i como sendo

$$\delta_i = (1 + \kappa\rho E)^2 f_i^{'}(u_i)\sum_k \delta_k w_{ki}. \tag{45}$$

VII. Exemplos Práticos

Com o objetivo de demonstrar o algoritmo de retropropagação modificado com as expressões (44) e (45), foram desenvolvidas duas aplicações práticas: a classificação de conjuntos bidimensionais, e classificação de dígitos manuscritos (OCR). Com relação à primeira aplicação, foi utilizado um exemplo de classificação de padrões simples de um conjunto de 300 pontos bidimensionais pré-classificados em três clusters distintos (veja figura 1). A idéia foi comparar não apenas o erro global em várias épocas de treinamento, mas também o número de classificações com erro (c_e) para as amostras do conjunto de treinamento. Para isso, foi utilizado um perceptron multicamada com algoritmo de treinamento modificado com a métrica riemaniana (indicado por MLP-MR). Para confrontar os resultados foi utilizado também um perceptron multicamada com algoritmo de retropropagação normal (indicado por MLP-NO). As redes MLP foram construídas com dois neurônios na camada de entrada, dois neurônios na camada oculta e três neurônios na camada de saída, totalizando 15 pesos e o espaço de pesos 16 dimensões. Durante o treinamento,

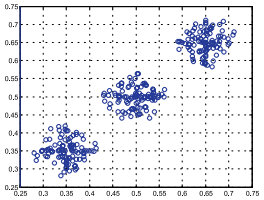

Fig. 1. Conjunto de pontos utilizados para o exemplo.

cada neurônio de saída era ativado com o valor "1" caso o ponto pertencesse à classe respectiva e o valor "0" às classes remanescentes. O algoritmo de retropropagação foi executado 100 vezes para cada rede, considerando o número de épocas variando de 1 até 20. Em cada uma, os 300 pontos foram apresentados às redes de forma *on-line*.

Os parâmetros de momento e aprendizado para MLP-NO e MLP-MR, e os parâmetros densidade inicial e fator de escala para MLP-MR constam na tabela 1.

Tabela 1: Parâmetros

Parâmetro	MLP-NO	MLP-MR
α	0,30	0,30
η	0,80	0,25
ρ	-	32,00
κ	-	1,00

A densidade espacial ρ decrescia a um fator de $(1-q)$, com $q = 0,003$, a cada época de treinamento, até se alcançar um limiar de erro $E_0 = 0,06$. Abaixo deste limiar, assumia-se $\rho = 0$ e $\eta = 0,8$, ou seja, o algoritmo retornava ao espaço de busca normal. Optou-se por manter o fator de escala κ constante durante todo o processo de treinamento.

Tabela 2: Comparação dos Resultados

Épocas	MLP-MR E	MLP-NO E	MLP-MR c_e	MLP-NO c_e
1	0,117443	0,335187	4,970297	159,6236
2	0,056281	0,143381	1,134328	60,40298
3	0,037165	0,064571	0,846906	34,52117
4	0,028168	0,030757	0,443128	17,04028
5	0,021633	0,018718	0,231054	7,857671
10	0,007401	0,006106	0,170829	0,743257
15	0,004203	0,003909	0,304464	0,342438
20	0,002881	0,002882	0,105947	0,075962

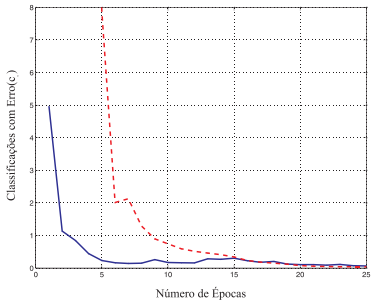

Fig. 2. Comparação do índice c_e entre as redes MLP-MR (linha sólida) e MLP-NO (linha pontilhada).

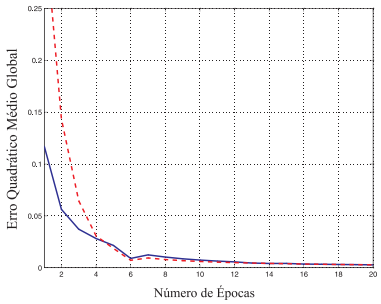

Fig. 3. Comparação do valor de erro global entre as redes MLP-MR (linha sólida) e MLP-NO (linha pontilhada).

A tabela 2 ilustra para cada época de treinamento o erro médio obtido para cada rede, bem com da média de classificações com erro (c_e) para cada uma, durante as 100 execuções de treinamento para cada época. Os resultados podem ser comparados também nos gráficos das figuras 2 e 3.

Os resultados evidenciados por esta tabela demonstram que o erro de classificação c_e para a rede MLP-MR é mínimo, apenas com uma época de treinamento. Na figura 4 podem ser comparadas três situações para cada tipo de MLP (onde pode-se notar até mesmo um comportamento não-linear na separação das classes). A partir de 20 épocas, o índice c_e das duas redes ficam no mesmo patamar. Ou seja, a rede MLP-MR em média começa com ótima convergência, chegando o erro global mínimo E num patamar onde a rede MLP-NO irá alcançar somente com um número maior de épocas. O algoritmo para MLP-MR começa assim com alto grau de curvatura, lançando a busca pelo mínimo bem próxima do valor ótimo.

Deve-se ressaltar também que (para uma época) em 4 das 100 execuções a rede MLP-MR não convergiu, resultando no c_e alto mostrado na tabela 2 (4,970297). Excluindo-se estas execuções do cálculo da média, o valor de c_e resultaria em 1,0833.

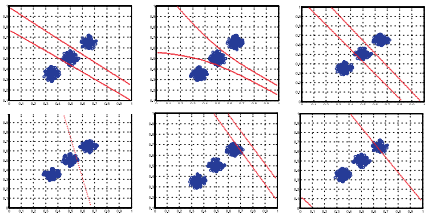

Fig. 4. Três exemplos de separação de classes com a rede MLP-MR (acima) e com a rede MLP-NO (abaixo) após uma época de treinamento.

O exemplo prático a seguir trata da utilização de um MLP para classificação de dígitos manuscritos (OCR) tal como descrito em [30][31]. A aplicação foi construída de forma a considerar imagens de 32x32 pixels como entrada. Um módulo pré-processador transforma a imagem de entrada através de uma camada de convolução produzindo imagens 32x32 pixels, uma camada de subamostragem reduzindo o tamanho para 16x16 pixels, e uma camada de subamostragem redundante que gerava por sua vez 9 campos receptivos de 8x8 pixels de imagens. No total, os campos receptivos geraram 576 pontos, que forma considerados como entradas da rede MLP. Como saída da rede MLP foram considerados 10 neurônios, um para cada dígito respectivo. Como não houve camadas ocultas, o total de pesos da rede MLP era de 5760. Na fase de treinamento, 100 amostras foram alimentadas (10 para cada dígito), e na camada de saída o valor '1' foi associado ao dígito respectivo, sendo '0' aos demais. Quando a fase de treinamento era executada, a cada época, o neurônio escolhido tendia a assumir o valor '1' e os outros, por sua vez, assumir o valor '0'.Como na aplicação anterior, duas redes foram utilizadas. O indicador utilizado para comparar o desempenho das redes MLP é o *desvio absoluto de classificação*, calculado pela diferença positiva entre o valor do neurônio do dígito correto e o somatório dos demais neurônios (O valor ideal é 1, indicando uma ótima classificação pela rede). Neste indicador pode se notar o bom desempenho da rede MLP-MR já no início do treinamento em relação à MLP-NO (figura 5).

Table 3: Parâmetros para aplicação de OCR

Parâmetro	MLP-NO	MLP-RM
α	0.40	0.40
η	0.95	0.95
ρ	-	25,00
κ	-	0.08

VIII. Conclusão

A utilização da métrica riemanniana para o espaço de pesos se mostrou bastante promissora. Além do aspecto de inovação, os resultados práticos obtidos aqui reforçam o formalismo desenvolvido. O aspecto mais relevante levantado por este trabalho é o posicionamento do vetor de busca próximo à solução final ainda nas primeiras épocas de treinamento. Entretanto,

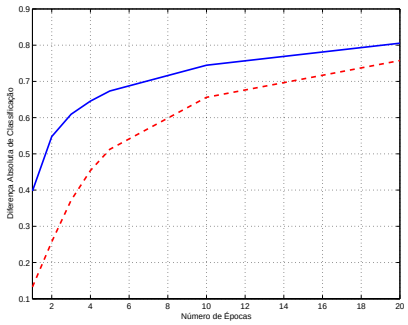

Fig. 5. Comparação da diferença absoluta de classificação entre a rede MLP-RM (linha sólida) e a rede MLP-NO (linha pontilhada).

um desenvolvimento mais profundo sobre a adoção da geometria riemanniana ao processo de convergência é necessária, tal como aplicação de outros tipos de soluções métricas e exemplos práticos, assim como estudos de impacto sobre pontos de mínimos locais. Um aspecto interessante a ser ressaltado no treinamento da rede MLP-MR é que, quando o conjunto de treinamento é relativamente grande, em cada apresentação de amostra (dentro da época de treinamento) a busca tende a se corrigir mais rapidamente em função do fator de curvatura presente no termo de correção de pesos modificado.

Estão previstos em trabalhos futuros a comparação da solução métrica obtida, utilizando algoritmos de retropropagação com uso de informação prévia de gradiente e técnicas de passo adaptativo, bem como métodos de segunda ordem imersos em espaço riemanniano. Um estudo comparativo para verificar o isomorfismo deste modelo com técnicas de passo adaptativo também será considerado.

REFERENCES

[1] S. Haykin. *Neural Networks - A Compreensive Foundation*. Macmillan Coll. Pub. Com. Inc. pp.1-41, 1994.

[2] S. R. Kulkarni, G. Lugosi & S. S. Venkatesh. Learning Pattern Classification - A Survey. *IEEE Transactions On Information Theory*, (4)6:2178-2205, 1998.

[3] P. J. Werbos. *Beyond regression: New tools for prediction and analisys in the behavioral sciences*. tese de doutorado, Harvard University, Cambridge, MA, 1974.

[4] D. E. Rumelhart, G. E. Hinton & R. J. Williams. *Learning representations of back-propagation errors*, em *Nature (London)*, 323:533-536., 1986.

[5] R. Battiti. First and second order methods for learning: Between steepest descent and newton´s method. *Neural Computation*, 4(2):141-166, 1992.

[6] W. L. Buntine & A. S. Weigend. Computing second derivatives in feedforward networks: A review. *IEEE Trans. on Neural Networks*, 5(3):480-488, 1993.

[7] S. E. Fahlman. *An Empirical Study of Learning Speed in Backpropagation Networks*. Technical Report, CMU-CS-88-162, 1988.

[8] M. Riedmiller and H. Braun, *A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm*, Proc. ICNN, San Fransisco, 1993.

[9] N. N. Schraudolph. Fast curvature matrix-vector products for second order gradient descent. *Neural Computation*, 14(7):1723-1738, 2002.

[10] L. B. Almeida, T. Langlois, J. D. Amaral e A. Plakhov. *Parameter adaptation in stochastic optimization*, 6:111-134. Cambridge University Press, 1999.

[11] G. B. Orr & T. K. Leen. Using curvature information for fast stochastic search. In M. I. Jordan et al editors, *Neural Information Processing Systems*. MIT Press, 9:606-612, Cambridge, 1996.

[12] G. P. Drago & S. Ridella. Statistically controled activation weight initialization (SCAWI). *IEEE Trans. on Neural Networks*, 5(6):989-993, 1994.

[13] D. Nguyen & B. Widrow. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptative weights. *Proc. of the Int. Joint Conference on Neural Networks*, 3(21):21-26. 1990.

[14] Y. F. Yam & T. W. S. Chow. A new method in determining the initial weights of feedforward neural networks. *Neural Computing*, 16(1):23-32, 1997.

[15] F. Biegler-Konig & F. Barnmann. A learning algorithm for multilayered neural networks based on linear least square problems. *Neural Networks*, 6:127-131, 1993.

[16] O. Fontella-Romero, D. Erdgomus, J. C. Principe, A. Alonzo-Betanzos e E. Castillo. Accelerating the convergence speed of neural networks learning methods using least squares. *European Symposium on Artificial Neural Networks*, pp.255-260, 2003.

[17] C. Charalambous. Conjugate gradient algorithm for efficient training of artificial neural networks. *IEEE Proceedings*, 139(3):301-310, 1992.

[18] M. Hagan & M. Menhaj. Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989-993, 1994.

[19] B. M. Wilamowski, S. Iplikçi, M.O. Efe and O. Kaynak. An Algorithm for Fast Convergence in Training Neural Networks, *Int. Joint Conf. On Neural Networks - IJCNN'01*, pp. 1778-1782, 2001.

[20] A. M. Chen, H. Lu & R. Hecht-Nielsen. On the geometry of feedforward neural network error surfaces. *Neural Computation*, 5(6):910-927, 1993.

[21] S. Phillips. The effect of representation on error surface. In P. Leong and M. Jabri, editors, *Fourth Australian Conference on Neural Networks (ACNN'93)*, pages 86–89, Australia, University of Sydney, 1993.

[22] Gallagher, M., Downs, T., Weight space learning trajectory visualization, em *Proc. Eighth Australian Conference on Neural Networks*, Melbourne, pp 55-59, 1997.

[23] C. Bishop. *Neural Networks for Pattern Recognition*. 2ed. Clarendon Press. Oxford, 1997.

[24] S. M. Carroll. *Lecture Notes on General Relativity*. Institute of Theoretical Physics. University of California, Santa Barbara. http://itp.ucsb.edu/ carrol/notes/, 1997.

[25] M. Berman & F. Gomide. *Cálculo Tensorial e Relatividade Geral - Uma Introdução*. 2.ed. Ed. McGraw-Hill. São Paulo, 1987.

[26] E. Butkov. *Física Matemática*. Editora Guanabara S.A. Rio de Janeiro, 1988.

[27] J. H. Heinbockel. *Introduction to Tensor Calculus and Continuum Mechanics*. Department of Mathematics and Statistics. Old Dominion University, 1996.

[28] G. Arfken. *Mathematical Methods for Physics* 3.ed. Academic Press, Inc. San Diego, 1985.

[29] K. F. Riley, M. P. Hobson, S. J. Bence. *Mathematical Methods for Physics And Engineering* 3.ed. Cambridge University Press. Cambridge, 2000.

[30] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel. Handwritten digit recognition with a back-propagation network, in *Advances in Neural Information processing Systems*, vol. 2, pp396-404, San Mateo, CA: Morgan Kaufmann, 1990.

[31] L. F. de Medeiros. *Redes Neurais em Delphi*. Florianópolis: Ed. Visualbooks, 2003.

Luciano Frontino de Medeiros nasceu em Porto Alegre-RS em 1966. É mestre em Informática pela Universidade Federal do Paraná, tendo cursado Administração e Física Licenciatura pela Universidade Federal de Santa Maria. É autor de livro e artigos na área de Redes Neurais e Reconstrução de Imagens de CT. Atualmente é professor de Inteligência Artificial, Banco de Dados e Sistemas de Informação nos cursos de graduação e pós-graduação do grupo UNINTER-PR.

Hamilton Pereira da Silva nasceu em Santa Maria-RS em 1964. É mestre em Informática Industrial pelo CEFET-PR e físico pela Universidade Federal do Paraná. Desenvolveu trabalhos e publicações na área de imagens de Tomografia Computadorizada. Atualmente é professor nos cursos de graduação e pós-graduação do grupo UNINTER-PR.