# $7^o$ Trabalho de Redes Neurais

Péricles Lopes Machado

**Resumo**—Neste trabalho, uma rede neural é treinada para simular a equação de recorrência:

$$x_k = x_{k-1} + \frac{ax_{k-s}}{1 + x_{i-s}^c} - bx_{k-1} + 0.1N, \quad (1)$$

onde $N$ é uma variável aleatória de distribuição normal com $\mu = 0$ e $\sigma^2 = 1$, $a = 0.2$, $b = 0.1$, $c = 10$ e $s = 17$.

## 1 A REDE NEURAL

A rede neural utilizada neste problema, com 7 neurônios na camada escondida, é apresentada na figura 1. Os neurônios na camada escondida possuem uma função de ativação sigmoidal e os da camada de saída possuem uma função de ativação linear.

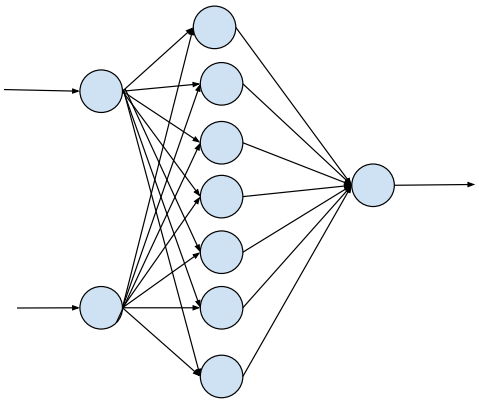

Figura 1. Rede neural utilizada.

Os pesos foram ajustados usando uma variável aleatória $X$ com distribuição gaussiana contida no intervalo $[-0.1, 0.1]$. A taxa de aprendizado utilizada foi de $\lambda = 0.01$ e a taxa de momento foi de $\alpha = 0.001$.

## 2 A GERAÇÃO DOS CONJUNTOS DE TREINAMENTO, VALIDAÇÃO E TESTE

Inicialmente, foram gerados $M = 10000$ amostra do sinal (1), no formato $v_i = (x_{i-1}, x_{i-s}, x_i)$, com $i = s+1, ..., M$, onde $x_n = 0$, para $n = 1, 2, ..., s$. O conjunto de amostras $S$ gerado foi divido em três partes $U_0, V, T$, onde $U_0$ contém $70\%$ do conjunto $S$ e é filtrado para se gerar o conjunto de treinamento, $V$ é o conjunto de validação e contém $20\%$ do conjunto $S$ e $T$ é o conjunto de teste e contém $10\%$ de $S$. A partição do conjunto $S$ em $U_0$, $V$ e $T$ foi realizada de forma aleatória.

Para tornar mais eficiente o treinamento, foi realizada uma filtragem pra eliminar dados de entrada com grande semelhança. A medida de semelhança entre duas amostras de entrada, $\vec{v}_i = (x_{i-1}, x_{i-s}, x_i)$ e $\vec{v}_k = (x_{k-1}, x_{k-s}, x_k)$, foi a função dada na equação(2).

$$D(\vec{v}_i, \vec{v}_k) = \sqrt{< \vec{d_{i,k}}, \vec{d_{i,k}} >}, \quad (2)$$

onde $\vec{d_{i,k}} = \vec{v}_i - \vec{v}_k$.

O algoritmo de filtragem é o apresentado a seguir:

- $\vec{x}_{ant} = (0, 0, 0)$.
- Define-se uma distância mínima $\epsilon$ que as amostras tem de ter com relação a $x_{ant}$ para entrar no conjunto de treinamento $U$.
- Para cada elemento $v_k$ de $U_0$, verifica-se o valor de $D(v_k, x_{ant})$, se este valor for maior que $\epsilon$, então $v_k$ entra para o conjunto $U$ e $x_{ant}$ passa ser igual a $v_k$.

A utilização desse algoritmo, com um $\epsilon = 0.1$, permitiu reduzir o conjunto de treinamento em $61.51\%$.

## 3 O TREINAMENTO

O treinamento foi realizado em 5 seções. Cada seção apresentava a seguinte divisão:

- O algoritmo *Aprendendo com a natureza* é utilizado com o conjunto de treino $U$ sendo apresentado no modo *batch*.
- O algoritmo *Aprendendo com a natureza* é utilizado com o conjunto de treino $U$ sendo apresentado no modo *online*.
- O algoritmo *backpropagation*, com o método dos momentos, é utilizado usando-se o mesmo conjunto de treino $U$ apresentado no modo *batch*.
- O algoritmo *backpropagation*, com o método dos momentos, é utilizado usando-se o mesmo conjunto de treino $U$ apresentado no modo *online*.
- O algoritmo *Aprendendo com a natureza* é utilizado com o conjunto de treino $U$ sendo apresentado no modo *batch*.
- O algoritmo *Aprendendo com a natureza* é utilizado com o conjunto de treino $U$ sendo apresentado no modo *online*.

Em cada seção, cada etapa executa 100 passos do algoritmo de treinamento e realiza uma validação utilizando-se o conjunto $V$. O algoritmo *Aprendendo com a natureza*

é importante, pois permite que o treinamento possa progredir mais iterações sem ficar preso rapidamente num mínimo local, graças à pertubação nos pesos que é realizada. Neste trabalho, o algoritmo *aprendendo com a natureza* utiliza uma variável aleatória $dW$ com distribuição uniforme definida no intervalo $[-0.01, 0.01]$. Já o *backpropagation* é importante para se localizar mais rapidamente um mínimo local dada uma configuração da rede gerada pelo *Aprendendo com a natureza*.

## 4 RESULTADOS

A seguir, são apresentados os gráficos da evolução do erro em cada uma das 5 seções de treinamento.

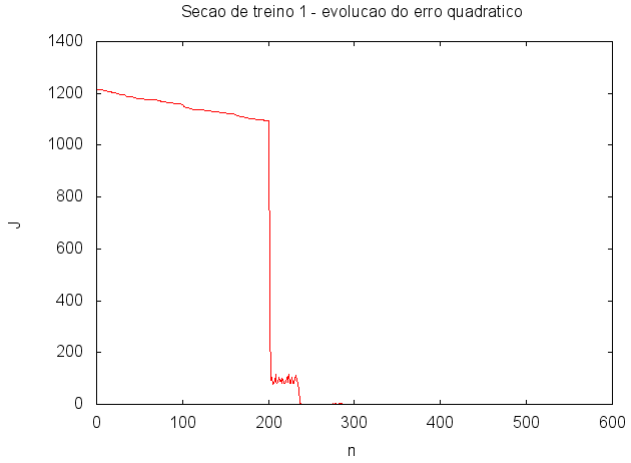


Figura 2. Evolução do erro da seção de treino 1.

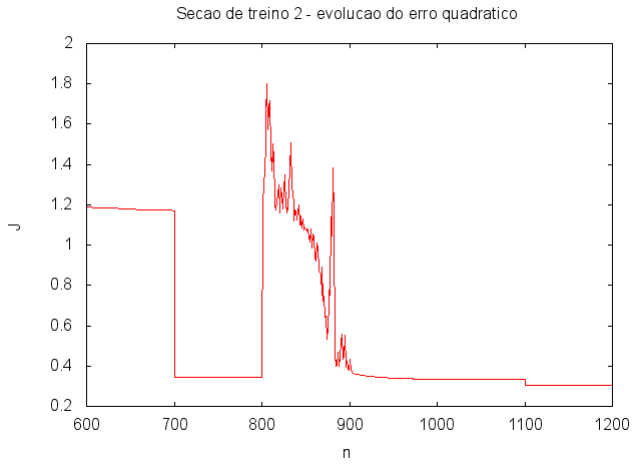


Figura 3. Evolução do erro da seção de treino 2.

Conforme pode-se ver na Fig. 2, inicialmente o algoritmo *backpropagation* é fundamental para que rapidamente se chega próximo à uma região de mínimo, já que o algoritmo *aprendendo com a natureza* apresenta mais dificuldade para encontrar um ajuste de peso que reduza o erro nos primeiros passos do algoritmo. Mas, o que começa a se notar a partir da seção de treino 2

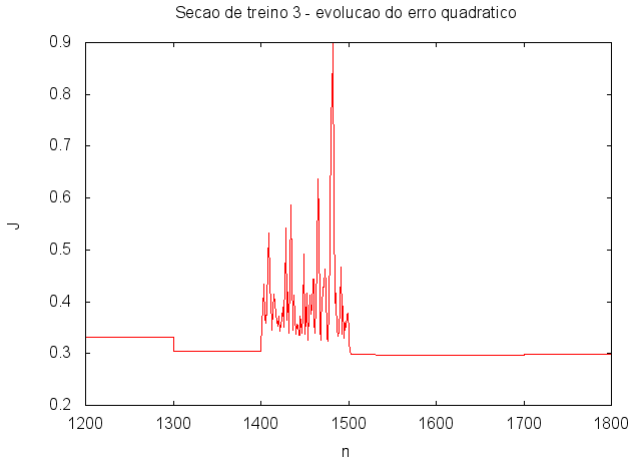


Figura 4. Evolução do erro da seção de treino 3.

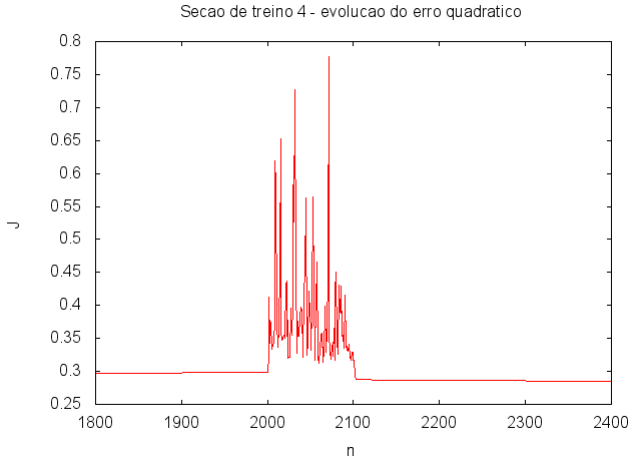


Figura 5. Evolução do erro da seção de treino 4.

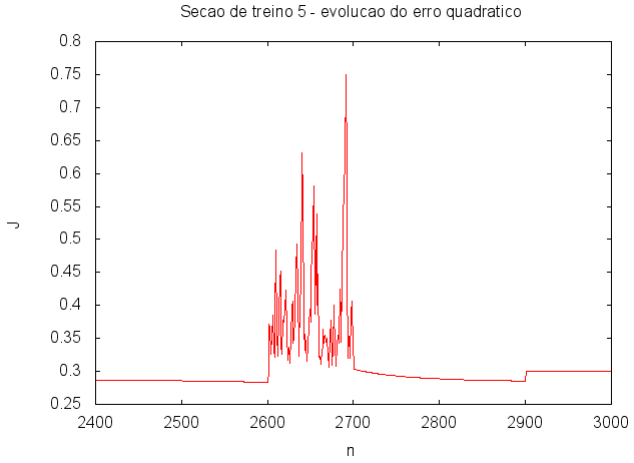


Figura 6. Evolução do erro da seção de treino 5.

(ver Fig. 3 a 6) é que o *aprendendo com a natureza* passa a ser fundamental pra se atingir novas reduções do erro, já que o *backpropagation* se estagna em mínimos locais.

Observando-se a evolução da validação em cada seção

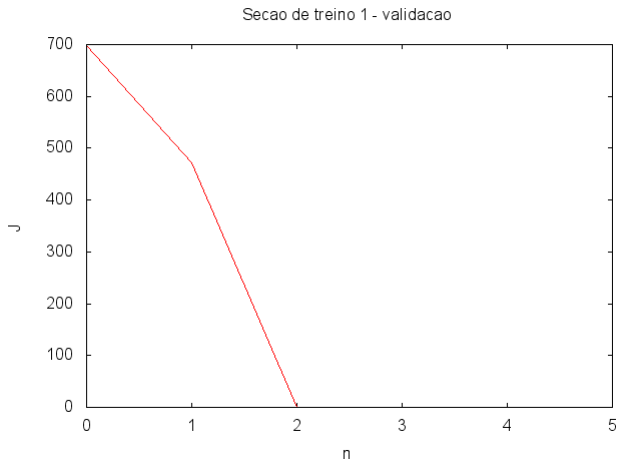


Figura 7. Evolução da validação em cada etapa da seção de treino 1.

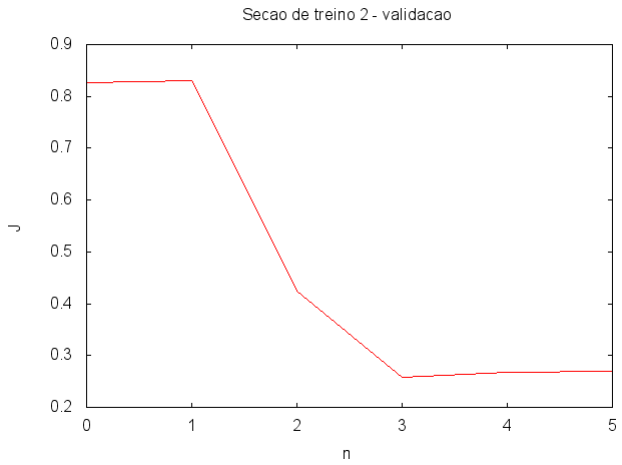


Figura 8. Evolução da validação da seção de treino 2.

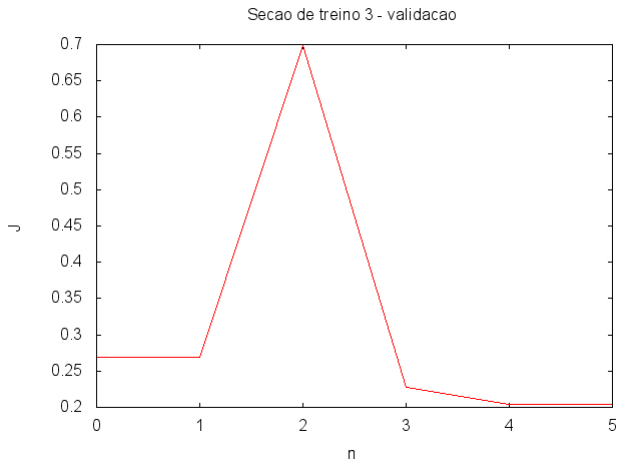


Figura 9. Evolução da validação da seção de treino 3.

de treinamento, também pode-se observar que o *aprendendo com a natureza* exerce papel fundamental para se conseguir reduções na função erro após a fase inicial do

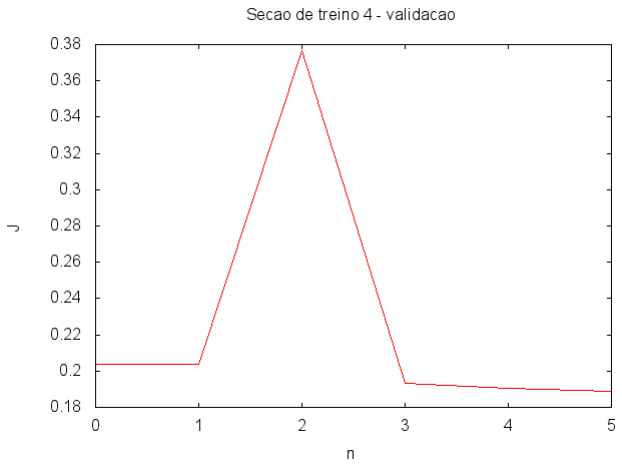


Figura 10. Evolução da validação da seção de treino 4.

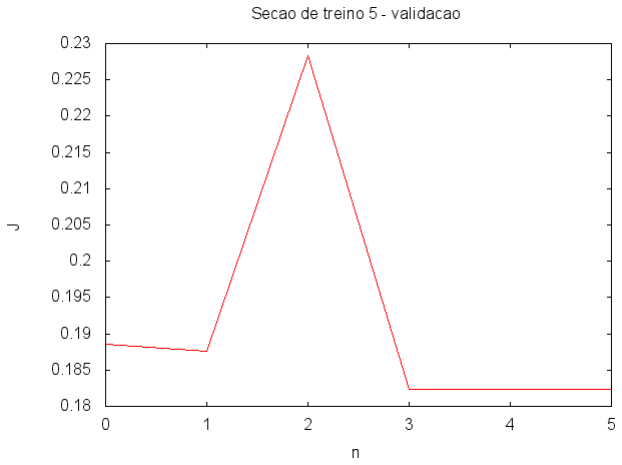


Figura 11. Evolução da validação da seção de treino 5.

treinamento.

Após realizado o treinamento, um teste foi feito utilizando-se o conjunto $T$ definido na seção 2 deste trabalho. O erro quadrádico obtido utilizando-se a rede neural este conjunto foi de $0.098$, e a curva obtida pode ser vista na Fig. 12

## 5 Conclusão

Neste trabalho, uma filtragem das amostras de uma série temporal para se gerar um conjunto de treinamento foi utilizada para tornar mais eficiente o ajuste de pesos de uma rede neural. Além disso, foram combinados dois algoritmos de ajuste, o *backpropagation* com momentos e o *aprendendo com a natureza*, para se conseguir uma maior minimização da função custo.

Como foi observado na seção 4, o *aprendendo com a natureza* foi fundamental para se obter maiores reduções no erro após as etapas iniciais do treinamento. Isso demonstra que utilizar pertubações aleatórias pode ser uma eficiente técnica para se obter maiores reduções na função custo. A Fig. 12 ilustra que esta técnica de treino

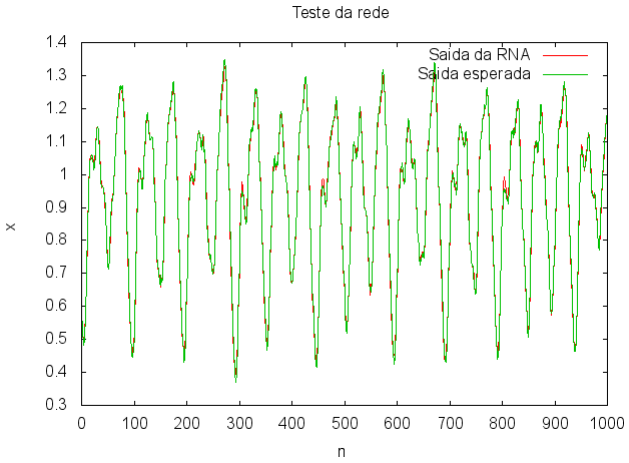


Figura 12. Teste para a RNA treinada neste trabalho.

obtém resultados satisfatórios no caso do treinamento da série temporal dada na equação (1).