oi

Renato Moura Martins Medeiros

6 de Outubro de 2016

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Capítulo 2

Preliminares

Uma *Rede Neural Artificial* (RNA) é um modelo computacional capaz de abstrair uma determinada informação dado um conjunto de exemplos dela. Esquematicamente, ela é representada na forma de um grafo [11] e é composta por unidades chamadas de *neurônios*, cujo objetivo principal é emular o funcionamento de um neurônio real, mesmo que de forma grosseira [7]. O *conhecimento* de uma rede neural artificial está no valor dos pesos que ligam seus neurônios, exceto em redes sem peso [1, 2], que não serão utilizadas nesse trabalho.

A larga utilização de redes neurais artificiais para a predição de valores em séries temporais [9, 8] se explica porque elas são ideais para esse propósito [5] e tem a capacidade de aproximar qualquer função, contínua ou não [3]. Para aumentar a acurácia na previsão da rede, recomenda-se pré-processar a série temporal com o objetivo de atenuar o seu ruído [4].

O uso de *onduletas*[1] é descrito por [15, 14] a fim de pré-processar séries temporais e, com isso, atenuar seu ruído. Onduletas são funções matemáticas que satisfazem certas condições e que são capazes de representar dados ou outras funções [6] tanto no eixo de frequências quanto no de tempo. Dentre as tantas onduletas existentes, a *Transformada de Haar* foi a escolhida para este trabalho porque, de acordo com [10], a transformada de Haar é matematicamente simples, converge rápido e possui alta precisão.

2.1 Nomenclatura

Seja y uma série temporal de Y elementos e y_i um elemento de y tal que $0 < i < Y$.

2.2 Trabalhos relacionados

[1]Embora a maioria dos autores prefira usar o termo em inglês *wavelet*, neste trabalho usamos o termo na língua portuguesa já que este foi o padrão escolhido nos demais termos usados no trabalho

Capítulo 3

Wavelets

Capítulo 4

Redes Neurais Artificiais

Redes neurais talvez sejam umas das ferramentas mais importantes na aprendizagem de máquina. Sua capacidade de aproximar qualquer função, mesmo as não-contínuas [3], tornam-nas importantes ferramentas para classificação e predição de séries.

Existem redes de aprendizado supervisionado, onde a rede precisa da resposta certa no seu treinamento, e de aprendizado não-supervisionado, onde esse valor não é necessário. O *treinamento* de uma rede perceptron de múltiplas camadas permite o ajuste dos pesos da rede de forma que ela responda a estímulos similares com um erro minimizado. Como o perceptron de múltiplas camadas se baseia em um aprendizado supervisionado, é necessário dispor do valor esperado como resposta da rede dada uma entrada.

4.1 Perceptron de Múltiplas Camadas

Em uma rede perceptron de múltiplas camadas, os neurônios estão contidos em conjuntos chamados *camadas*. Esquematicamente, as camadas são perfiladas paralelamente de forma que a primeira recebe o nome de *camada de entrada* e a última de *camada de saída*. Toda e qualquer camada que esteja entre a camada de entrada e a camada de saída é chamada de *camada intermediária* ou, mais comumente, *camada oculta*. A Figura 4.1 apresenta um exemplo de uma rede perceptron de múltiplas camadas.

Cada neurônio pertence unicamente a uma camada e se conecta a todos os neurônios da camada seguinte. Essas conexões possuem um valor atribuído, chamado de *peso*. Na Figura 4.1 cada neurônio é representado por um círculo amarelo, enquanto cada peso é representado por $P_{i,j}$.

O interior de um neurônio possui duas funções, a *função de agrupamento* e a *função de ativação*, também chamada de *função de transição* ou *função de propagação*. Excetuando-se os neurônios na camada de entrada, os demais recebem vários estímulos, como o resultado dos neurônios da camada anterior, bem como os pesos de conexão. A função de agregação de um neurônio soma o produto da saída dos neurônios da camada anterior com seus respectivos pesos de conexão, conforme Equação 4.1.

$$f_{ag} = \sum_{n=0}^{N} P_n S_n \tag{4.1}$$

Na Equação 4.1, f_{ag} é a função de agregação de um neurônio qualquer, N é o número de neurônios na camada anterior, P_n é o peso de conexão entre o neurônio da camada anterior e o atual e S_n é o valor calculado pelo neurônio da camada anterior. Imaginar P como um vetor de pesos e S como um vetor de valores calculados pelos neurônios da camada anterior permite reescrever a Equação 4.1 conforme Equação 4.2, onde P^T é o vetor transposto de P.

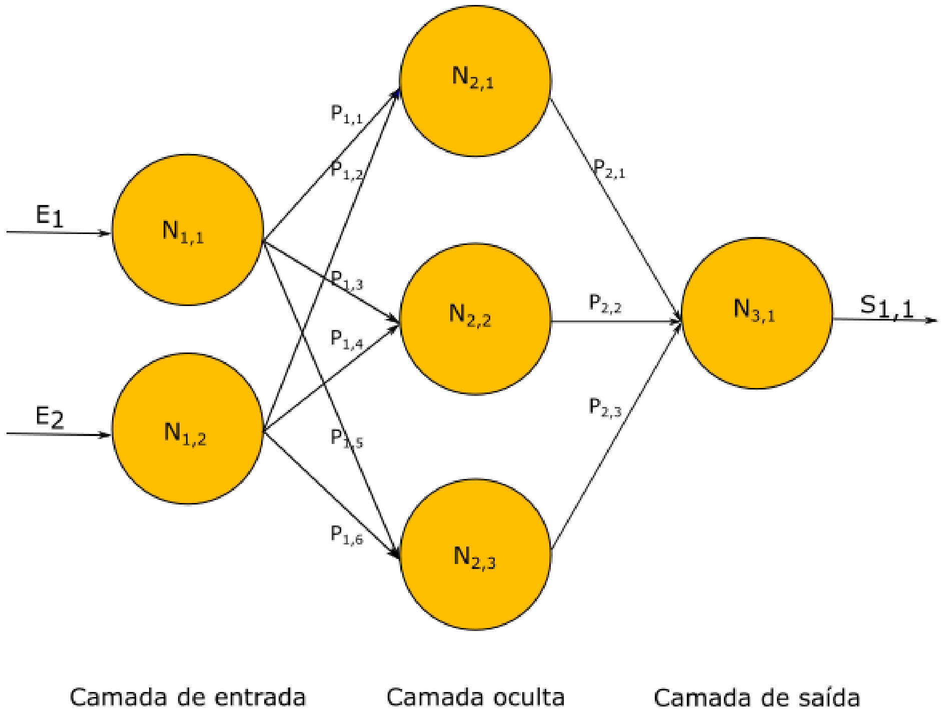

Figura 4.1: Um Perceptron de Múltiplas Camadas com três camadas. Dois neurônios na primeira, três na segunda e apenas um na terceira.

$$f_{ag} = P^T.S \tag{4.2}$$

Já a função de transição recebe o resultado da função de agregação e o usa para calcular a saída do neurônio. A Figura 4.2 esquematiza o funcionamento interno de um neurônio. O neurônio da Figura 4.2 recebe a resposta de dois neurônios da camada anterior, bem como o peso de conexão entre cada um desses neurônios e o atual.

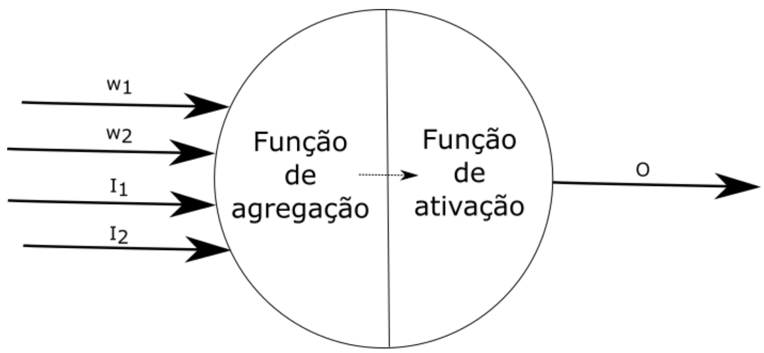

Figura 4.2: Funcionamento interno de um neurônio, que recebe estímulo de dois neurônios da camada anterior.

Diferente do que acontece com a função de agregação, existem diversas funções de ativação que podem ser usadas em um perceptron de múltiplas camadas (ENCONTRAR UM TRABALHO QUE FAÇA UM ESTUDO COMPARATIVO ENTRE ESSAS FUNÇÕES, OU QUE USE MAIS DE UMA FUNÇÃO PARA CITAR AQUI). Inclusive, a última camada da rede pode ter uma função de transferência diferente das camadas anteriores, geralmente para normalizar a saída da rede [13].

4.2 O Algoritmo de Retropropagação

O algoritmo de retropropagação[1] foi proposto por ENCONTRAR ARTIGO DE QUEM PROPÔS E COLOCAR AQUI e tem por objetivo modificar os pesos da rede de forma que a diferença entre o valor de saída da rede e o valor real, também chamado de *erro de predição*, seja o menor possível. Ele se baseia no uso de métodos de otimização, como o método do gradiente.

O treinamento de uma rede perceptron de múltiplas camadas precisa de dois vetores de valores, um de entrada e outro de saída. O algoritmo possui duas grandes etapas; a alimentação da rede e a retropropagação do erro.

Passo 1: Alimentação da rede

O vetor de entrada é o estímulo a ser passado para a rede; ele tem o mesmo tamanho da camada de entrada. Dessa forma, cada valor do vetor é passado diretamente para um neurônio, sem utilizar a função de agregação. A Figura 4.3 (retirada de [12]) ilustra como isso é feito esse processo.

A saída de um neurônio é o valor da função de ativação nele presente. Se o neurônio estiver presente na camada de entrada ou em uma camada oculta, esse valor é passado a todos os neurônios da camada seguinte, cujas funções de agregação calculação o valor a ser passado para a sua função de ativação. Se o neurônio estiver presente na camada de saída, o valor calculado pelo neurônio será a resposta da rede naquele neurônio.

Passo 2: Retropropagação do erro

Uma vez terminado o passo 1, a última camada da rede possui suas respostas para o estímulo passado. Tais respostas precisam ser comparadas com o vetor de saída, que possui as respostas esperadas pela rede. Dessa forma, o vetor de saída precisa ser do mesmo tamanho da camada de saída da rede.

A função de erro mais comum a ser utilizada para comparar a resposta da rede com o vetor de saída ENCONTRAR ARTIGO QUE DIGA ISSO é o erro quadrático (Equação 4.3, onde i é índice de um neurônio na camada de saída, y_i é o valor esperado para este neurônio e S_i é o valor que a rede respondeu.

$$E = \frac{1}{2}\sum_i (y_i - S_i)^2 \tag{4.3}$$

Deseja-se saber o quanto modificar o peso $P_{i,j}$ na camada de saída modifica o erro E (Equação 4.3). Matematicamente, isso pode ser expressado pela Equação 4.4. A Equação 4.5 possui a expansão através da aplicação da regra da cadeia.

$$\frac{\partial E}{\partial P_{i,j}} \tag{4.4}$$

$$\frac{\partial E}{\partial P_{i,j}} = \frac{\partial E}{\partial S_{i,j}} * \frac{\partial S_{i,j}}{\partial E_{i,j}} * \frac{\partial E_{i,j}}{\partial P_{i,j}} \tag{4.5}$$

O primeiro termo da Equação 4.4

4.2.1 Um exemplo numérico

[1] Também chamado por alguns autores de *propagação de retorno do erro*

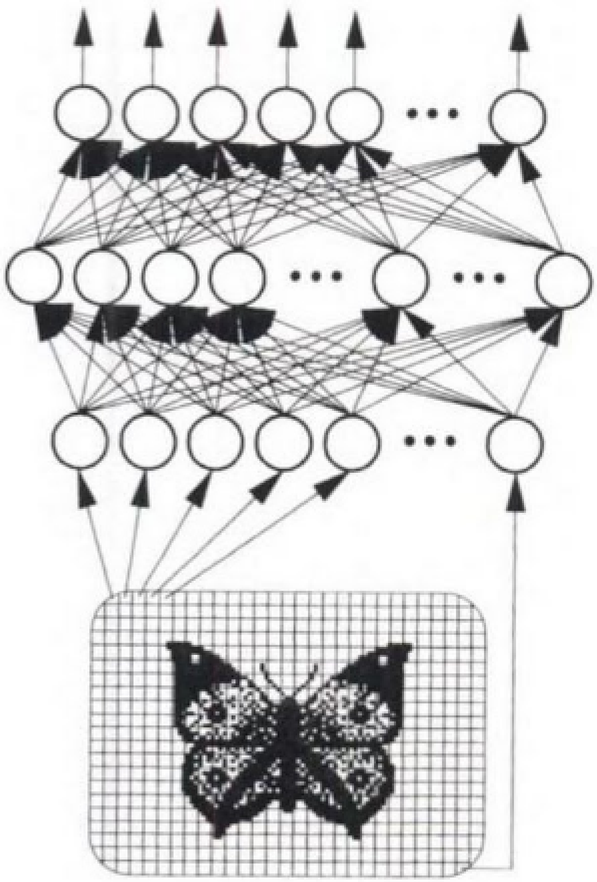

Figura 4.3: Ilustração retirada de [12] onde o valor de cada pixel é enviado para um neurônio na camada de entrada

Capítulo 5

Proposta de Trabalho

Bibliografia

[1] I. Aleksander, W. Thomas, and P. Bowden. Wisard·a radical step forward in image recognition. *Sensor Review*, 4(3):120–124, 1984.

[2] W. W. Bledsoe and I. Browning. Pattern recognition and reading by machine. In *Papers Presented at the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '59 (Eastern), pages 225–232, New York, NY, USA, 1959. ACM.

[3] B. de Assis Pio. *Uma Introdução às Redes Neurais Artificiais para Estudos em Ecologia*. Bruno Luiz de Assis Pio, 2009.

[4] J. Gao, H. Sultan, J. Hu, and W. W. Tung. Denoising nonlinear time series by adaptive filtering and wavelet shrinkage: A comparison. *IEEE Signal Processing Letters*, 17(3):237–240, March 2010.

[5] C. R. Gent and C. P. Sheppard. Predicting time series by a fully connected neural network trained by back propagation. *Computing Control Engineering Journal*, 3(3):109–112, May 1992.

[6] A. Graps. An introduction to wavelets. *IEEE Computational Science and Engineering*, 2(2):50–61, Summer 1995.

[7] K. Gurney. *An Introduction to Neural Networks*. Taylor & Francis, 2003.

[8] J. M. Kuo, J. C. Principle, and B. de Vries. Prediction of chaotic time series using recurrent neural networks. In *Neural Networks for Signal Processing [1992] II., Proceedings of the 1992 IEEE-SP Workshop*, pages 436–443, Aug 1992.

[9] S. S. Rao, S. Sethuraman, and V. Ramamurti. A recurrent neural network for nonlinear time series prediction-a comparative study. In *Neural Networks for Signal Processing [1992] II., Proceedings of the 1992 IEEE-SP Workshop*, pages 531–539, Aug 1992.

[10] A. Reddy, S. Manjula, C. Sateesha, and N. Bujurke. Haar wavelet approach for the solution of seventh order ordinary differential equations. *MMEP*, 3(2):108–114, jun 2016.

[11] D. Skapura. *Building Neural Networks*. ACM Press Series. ACM Press, 1996.

[12] D. Skapura. *Building Neural Networks*. ACM Press Series. ACM Press, 1996.

[13] P. Swietojanski and S. Renals. Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 171–176, Dec 2014.

[14] L. Wang, K. K. Teo, and Z. Lin. Predicting time series with wavelet packet neural networks. In *Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on*, volume 3, pages 1593–1597 vol.3, 2001.

[15] Q. Zhang and A. Benveniste. Wavelet networks. *IEEE Transactions on Neural Networks*, 3(6):889–898, Nov 1992.