

Redes Neurais



Cruzeiro do Sul Virtual
Educação a distância

Material Teórico



Processo de *Backpropagation*

Responsável pelo Conteúdo:

Prof. Dr. Alberto Messias da Costa Souza

Revisão Textual:

Prof. Me. Luciano Vieira Francisco

UNIDADE

Processo de *Backpropagation*



- Entendendo o Processo de *Backpropagation*;
- O Processo de Execução na Rede.



OBJETIVO DE APRENDIZADO

- Introduzir o conceito e o processo de aprendizagem por *backpropagation* ou retropropagação em redes neurais.



Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:



Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

Entendendo o Processo de *Backpropagation*

O processo *Backpropagation* é um dos principais meios de aprendizagem em uma rede neural. A principal ideia é ajustar os pesos de cada neurônio artificial usando essencialmente o valor esperado para a saída da rede e o erro ou distância da saída da rede para o valor esperado.

Para isso utilizamos a medida clássica que é a soma dos erros quadrados, comumente utilizada em outros algoritmos para análises de dados ou mesmo em validações de resultados de algoritmos.

Para relembrar o cálculo do erro podemos usar o seguinte exemplo:

Digamos que a saída esperada da rede seja o valor número de 9, porém a saída da rede retornou o valor de 7,5, observe que a distância entre 9 e 7,5 é 1,5. Este então seria um cálculo de erro para o dado valor de saída para o valor esperado. Comumente utilizamos esta lógica em diversos algoritmos, segue a formulação matemática para o cálculo da soma dos erros quadrados.

$$\sum_{i=1}^n (\bar{x} - x_i)^2$$

Figura 1 – Fórmula matemática da soma dos erros quadrados

Nesse caso observamos que pode existir um conjunto de saídas e o valor do erro total será dado pelo somatório, indo de 1 até N, de $X - X_i$, ou seja, o valor esperado subtraindo o valor retornado pela rede, elevado ao quadrado. O retorno deste erro será a distância total entre todos os valores esperados e valores retornados pela rede neural.

Este erro quadrado será um valor importante para retropropagar o erro para as outras camadas da rede ou para todos os outros neurônios. Note que é importante ainda saber qual é a importância do peso de todos os neurônios para o resultado de saída. Para este cálculo de “importância” ou peso do neurônio utilizaremos a derivada parcial de cada uma das funções de cada neurônio e a derivada parcial da função de erro.

Cabe destacar que você não deve se preocupar com o cálculo de uma derivada parcial, pois não necessariamente tivemos o contato com cálculo diferencial e integral em nosso curso.

Esta função de ativação ou a derivada parcial de cada neurônio é conhecida também como declínio de gradiente da rede ou de cada neurônio e a cada iteração procura-se sempre reduzir o erro de saída da rede neural.

Após repetidas vezes se propagando o erro da rede para trás ou para os neurônios ou camadas anteriores, o peso de cada neurônio vai se ajustando até chegar ao resultado esperado para a rede, quando isto ocorrer a rede estará treinada e poderá ser colocada em prática.

Dada esta introdução a respeito do conceito relacionados ao algoritmo *Backpropagation* vamos defini-lo e exercitá-lo.

Seguem sucintamente os passos do algoritmo:

- Inicializar pesos para os parâmetros que queremos treinar;
- Propagar para frente através da rede para obter os valores de saída;
- Definir o erro ou função de custo e seus primeiros valores de derivações;
- Retropropagar através da rede para determinar os derivados de erro;
- Atualizar as estimativas de parâmetro usando a derivada de erro e o valor atual.

Vamos ver a execução do algoritmo num exemplo prático!

Vamos usar uma rede neural com duas entradas, dois neurônios ocultos, dois neurônios de saída. Além disso, os neurônios ocultos e de saída incluirão um viés.

Aqui está a estrutura básica:

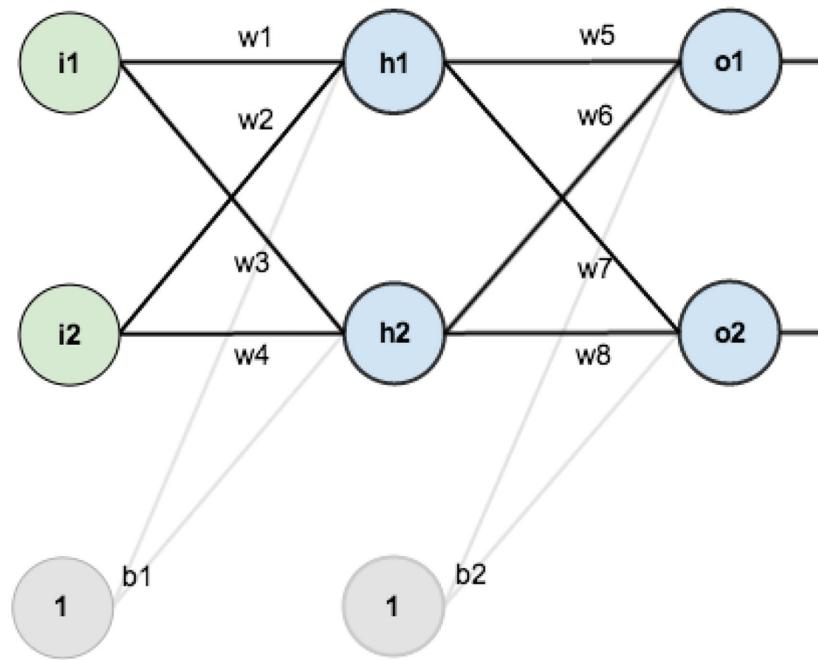


Figura 2 – Rede neural de exemplo

Fonte: Semanticscholar.org

Para termos alguns números com os quais trabalhar, aqui estão os pesos iniciais, os bias e entradas/saídas de treinamento:

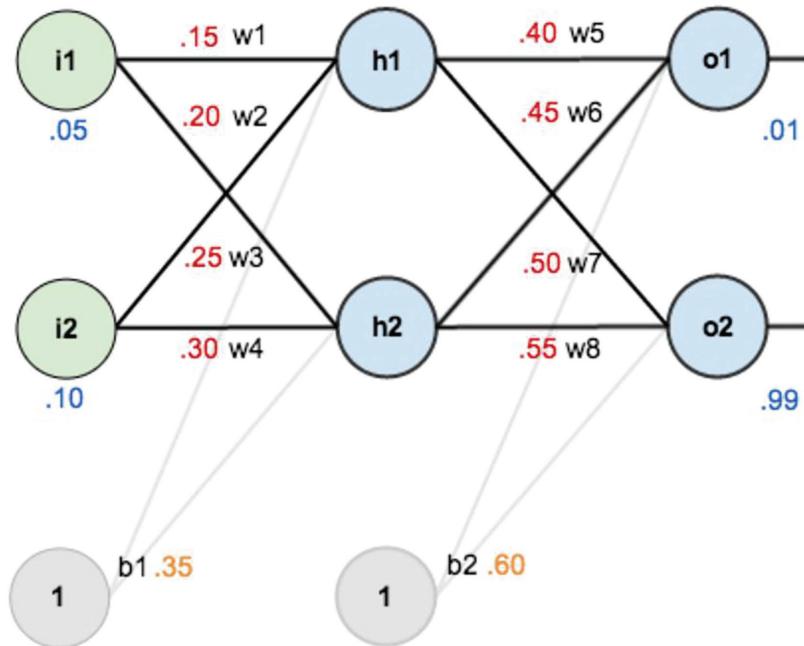


Figura 3 – Rede neural de exemplo com os valores definidos

Fonte: icl.utk.edu

O objetivo da retropropagação é otimizar os pesos para que a rede neural possa aprender como mapear corretamente entradas arbitrárias em saídas.

Trabalharemos com um único conjunto de treinamento: dados as entradas 0,05 e 0,10, queremos que a rede neural produza 0,01 e 0,99.

0 Processo de Execução na Rede

Para começar, vamos ver o que a rede neural prevê atualmente, dados os pesos e vieses acima e entradas de 0,05 e 0,10. Para fazer isso, alimentaremos essas entradas pela rede.

Calculamos a entrada líquida total para cada neurônio da camada oculta, somando a entrada líquida total usando uma função de ativação (aqui usamos a função logística ou LOG na base “Euler”) e, em seguida, repetimos o processo com os neurônios da camada de saída.

Veja como calculamos a entrada líquida total para h_1 ou a saída para o neurônio 1:

$$\begin{aligned} net_{h_1} &= w_1 \cdot i_1 + w_2 \cdot i_2 + b_1 \cdot 1 \\ net_{h_1} &= 0.15 \cdot 0.05 + 0.2 \cdot 0.1 + 0.35 \cdot 1 = 0.3775 \end{aligned}$$

Em seguida, o esmagamos usando a função logística para obter a saída de h_1 , ou executamos uma função de ativação, neste caso logística:

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

Realizando o mesmo processo para h_2 obter:

$$out_{h2} = 0.596884378$$

Repetimos esse processo para os neurônios da camada de saída, usando a saída dos neurônios da camada oculta como entradas.

Esta é a saída para O_1 , ou *output* 1:

$$\begin{aligned} net_{o1} &= w_5 \cdot out_{h1} + w_6 \cdot out_{h2} + b_2 \cdot 1 \\ net_{o1} &= 0.4 \cdot 0.593269992 + 0.45 \cdot 0.596884378 + 0.6 \cdot 1 = 1.1059905967 \\ out_{o1} &= \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.1059905967}} = 0.751365507 \end{aligned}$$

E realizando o mesmo processo para O_2 obtermos:

$$out_{o2} = 0.772928465$$

Calculando o Erro Total

Agora podemos calcular o erro para cada neurônio de saída usando a função de erro quadrático e somá-los para obter o erro total:

$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

Note que *target* é o valor esperado e *output* é a saída da rede neural.

Por exemplo, a saída alvo para O_1 é 0,01, mas a saída da rede neural é 0,75136507, portanto, seu erro é:

$$E_{o1} = \frac{1}{2} (target_{o1} - output_{o1})^2 = \frac{1}{2} (0.01 - 0.75136507)^2 = 0.274811083$$

Repetindo esse processo O_2 (lembrando que a meta é 0,99), obtemos:

$$E_{o2} = 0.023560026$$

O erro total da rede neural é a soma desses erros:

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

Processo Backpropagation

Nosso objetivo com a retropropagação é atualizar cada um dos pesos na rede de forma que eles façam com que a saída real fique mais próxima da saída alvo, minimizando assim o erro para cada neurônio de saída e a rede como um todo.

Camada de Saída

Considere w_5 . Queremos saber o quanto uma mudança em w_5 afeta o erro total, também conhecido como $\frac{\partial E_{total}}{\partial w_5}$.

Neste caso o $\frac{\partial E_{total}}{\partial w_5}$ é tido como a “a derivada parcial de E_{total} em relação a w_5 ”, ou em algumas referências pode-se chamar de gradiente em relação a w_5 “, conforme o mencionado no inicial da unidade.

Ao aplicar a regra da cadeia, sabemos que:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} \cdot \frac{\partial out_{o1}}{\partial net_{o1}} \cdot \frac{\partial net_{o1}}{\partial w_5}$$

O que seria a retropropagação do peso do neurônio w_5 , note que aqui estamos usando a derivada parcial, conforme o mencionado anteriormente.

Visualmente, aqui está o que estamos fazendo:

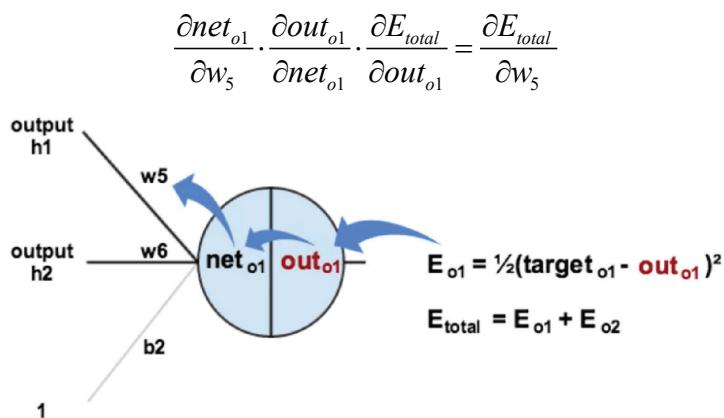


Figura 4
Fonte: Semanticscholar.org

Precisamos descobrir cada peça nesta equação.

Em primeiro lugar, quanto o erro total muda em relação à saída?

$$E_{total} = \frac{1}{2}(target_{o1} - output_{o1})^2 + \frac{1}{2}(target_{o2} - output_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 \cdot \frac{1}{2}(target_{o1} - output_{o1})^{2-1} \cdot -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - output_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

Aqui aplicamos a devirada da função de erro e também da função de ativação do neurônio, que são as duas primeiras linhas de equação.

Em seguida podemos calcular qual é a importância ou o quanto da saída varia com a variação deste neurônio.

Para isso usamos a derivada da função de ativação do neurônio, ou seja, a derivada parcial da função logística é a saída multiplicada por 1 menos a saída:

$$out_{o1} = \frac{1}{1 + e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

Finalmente, a quanto corresponde a entrada líquida total de o_1 mudança w_5 ?

$$net_{o1} = w_5 \cdot out_{h1} + w_6 \cdot out_{h2} + b_2 \cdot 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 \cdot out_{h1} \cdot w_5^{(l-1)} + 0 + 0 = out_{h1} = 0.593269992$$

Juntando tudo:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} \cdot \frac{\partial out_{o1}}{\partial net_{o1}} \cdot \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 \cdot 0.186815602 \cdot 0.593269992 = 0.082167041$$

Neste caso então usando os valores extraídos da função derivada parcial de cada uma das funções. Em algumas referências você poderá apenas encontrar o símbolo delta, para se referir à essa função de retropropagação.

Para diminuir o erro, subtraímos esse valor do peso atual (opcionalmente, multiplicado por alguma taxa de aprendizado, delta, que definiremos como 0,5):

$$w_5^+ = w_5 - \eta \cdot \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 \cdot 0.082167041 = 0.35891648$$

Podemos repetir esse processo para obter os novos pesos w_6 , w_7 e w_8 :

$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$

Realizamos as atualizações reais na rede neural após termos os novos pesos levando aos neurônios da camada oculta (ou seja, usamos os pesos originais, não os pesos atualizados, quando continuamos o algoritmo de retropropagação abaixo).

A seguir, vamos continuar os passos para trás para calcular novos valores para w_1 , w_2 , w_3 e w_4 .

Em outras palavras, precisamos calcular:

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} \cdot \frac{\partial out_{h1}}{\partial net_{h1}} \cdot \frac{\partial net_{h1}}{\partial w_1}$$

Ou visualmente:

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} \cdot \frac{\partial out_{h1}}{\partial net_{h1}} \cdot \frac{\partial net_{h1}}{\partial w_1}$$

↓

$$\frac{\partial E_{total}}{\partial out_{h1}} \cdot \frac{\partial E_{o1}}{\partial out_{h1}} \cdot \frac{\partial E_{o2}}{\partial out_{h1}}$$

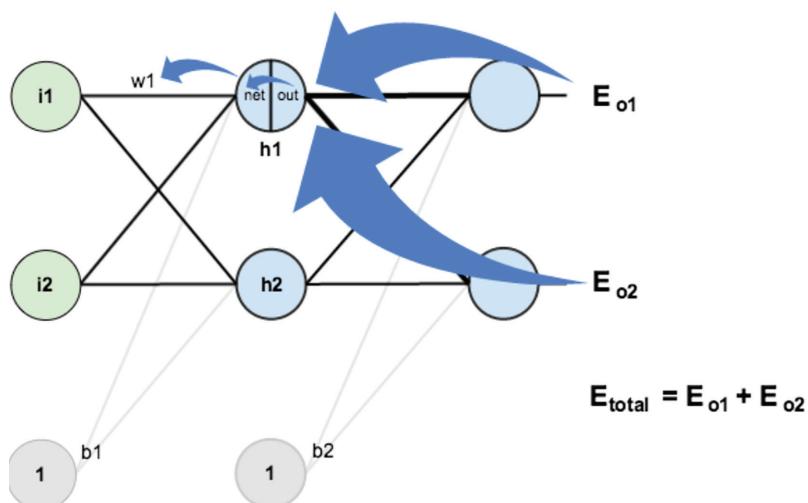


Figura 5
Fonte: Users.cselabs.umn.edu

Usaremos um processo semelhante ao que fizemos para a camada de saída, mas um pouco diferente para explicar o fato de que a saída de cada neurônio da camada oculta contribui para a saída (*e*, portanto, o erro) de vários neurônios de saída. Sabemos que isso out_{h1} afeta a ambos out_{o1} e out_{o2} portanto, é necessário levar em consideração seu efeito nos dois neurônios de saída:

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

Começando com $\frac{\partial E_{o1}}{\partial out_{h1}}$:

Podemos calcular $\frac{\partial E_{o1}}{\partial net_{o1}}$ usando valores que calculamos anteriormente:
 E $\frac{\partial net_{o1}}{\partial out_{h1}}$ é igual a w_5 :

$$net_{o1} = w_5 \cdot out_{h1} + w_6 \cdot out_{h2} + b_2 \cdot 1$$

$$\frac{\partial net_{o1}}{\partial out_{h1}} = w_5 = 0.40$$

Conectando-os:

Seguindo o mesmo processo para $\frac{\partial E_{o2}}{\partial out_{h1}}$, obtemos:

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.01904$$

Portanto:

Agora que temos $\frac{\partial E_{total}}{\partial out_{h1}}$, precisamos descobrir $\frac{\partial out_{h1}}{\partial net_{h1}}$ e, $\frac{\partial net_{h1}}{\partial w}$ em seguida, para cada peso:

$$out_{h1} = \frac{1}{1 + e^{-net_{h1}}}$$

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.59326999(1 - 0.59326999) = 0.241300709$$

Calculamos a derivada parcial da entrada líquida total em h_1 relação ao w_1 mesmo que fizemos para o neurônio de saída:

$$net_{h1} = w_1 \cdot i_1 + w_3 \cdot i_2 + b_1 \cdot 1$$

$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

Juntando tudo:

$$\frac{\partial E_{total}}{\partial w_1} = 0.36350306 \cdot 0.241300709 \cdot 0.05 = 0.000438568$$

Agora podemos atualizar w_1 :

$$w_1^+ = w_1 - \eta \cdot \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 \cdot 0.000438568 = 0.149780716$$

Repetindo isso por w_2 , w_3 e w_4

$$w_2^+ = 0.19956143$$

$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$

Finalmente, atualizamos todos os nossos pesos. Quando alimentamos as entradas 0,05 e 0,1 originalmente, o erro na rede era 0,298371109. Após esta primeira rodada de retropropagação, o erro total caiu para 0,291027924. Pode não parecer muito, mas depois de repetir esse processo 10.000 vezes, por exemplo, o erro cai para 0,0000351085. Neste ponto, quando alimentamos 0,05 e 0,1, os dois neurônios de saída geram 0,015912196 (vs 0,01 alvo) e 0,984065734 (vs 0,99 alvo).

Sendo assim, repetimos isso várias vezes até que o erro diminua e as estimativas dos parâmetros se estabilizem ou converjam para os valores esperados. Obviamente, não faremos todos esses cálculos manualmente e sim através da implementação do algoritmo *backpropagation*.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:



Livros

Inteligência Artificial

NORVIG, P. **In inteligência Artificial**. Grupo GEN, 2013. Disponível em sua biblioteca na área do aluno.



Leitura

Backpropagation Step by Step

<https://bit.ly/3lFDyeC>

Backpropagation example with numbers step by step

<https://bit.ly/3llISz1>

Redes Neurais Artificiais: Algoritmo Backpropagation

<https://bit.ly/3lKP9sM>

Referências

BISPO, F. M. D. S. M. L. L. P. H. C. F. S. C. **Inteligência artificial**. Grupo A, 2019. 9788595029392. Disponível em: <<https://integrada.minhabiblioteca.com.br/#/books/9788595029392/>>. Acesso em: 24/01/2020.

DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern classification**, 2nd Edition, Wiley, 2000, ISBN 978-0-471-05669-0.

GOODFELLOW, I.; BENGIO, Y; COURVILLE, A. **Deep Learning**, MIT Press, 2016. Disponível em: <<http://www.deeplearningbook.org>>. Acessado em 21/11/2019.

NORVIG, P. **Inteligência Artificial**: Grupo GEN, 2013. 9788595156104. Disponível em: <<https://integrada.minhabiblioteca.com.br/#/books/9788595156104/>>. Acesso em: 24/01/2020.

THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern Recognition**, Fourth Edition, 4th. ed. Academic Press, Inc., USA, 2008.



Cruzeiro do Sul
Educacional