

Relatório 13 - Redes Neurais

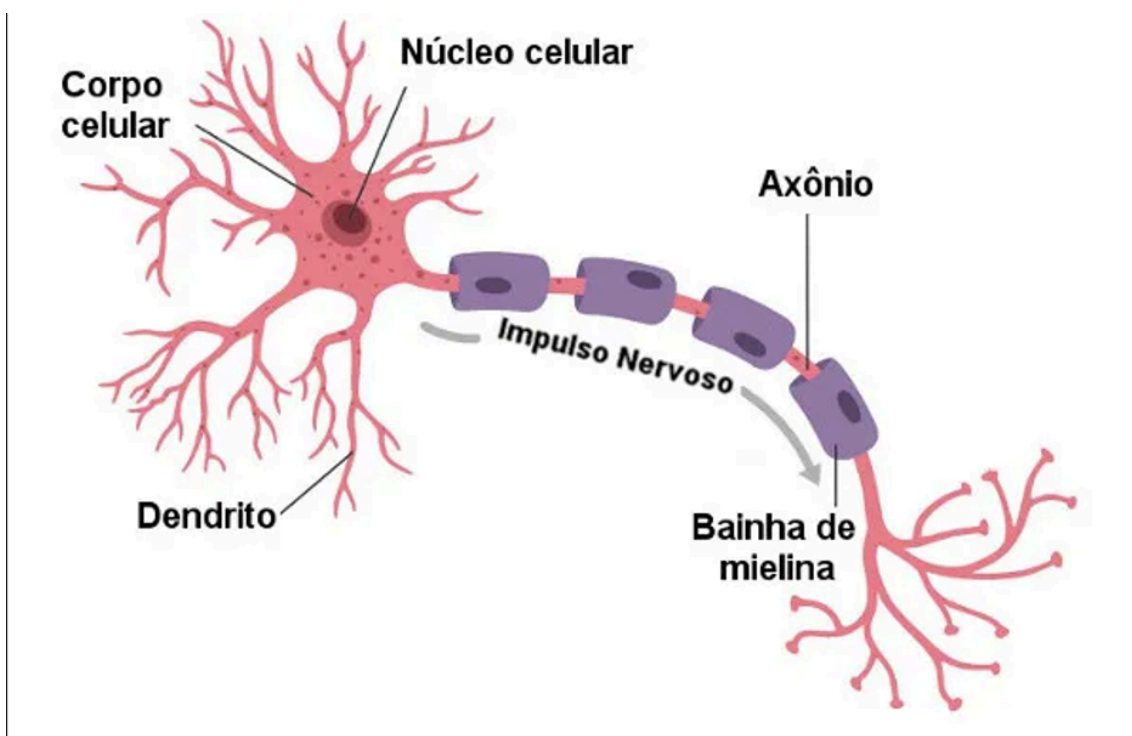
Lucas Scheffer Hundsdorfer

Descrição da atividade

Seção 3 - Teoria resumida sobre redes neurais artificiais

Aula 7 - Fundamentos Biológicos.

A aula começa explicando que dentro do nosso cérebro existem bilhões de neurônios e eles e suas conexões que são responsáveis pelas nossas habilidades, andar, ler, ver, falar com tudo que você é capaz de fazer.



Os dendritos é onde as informações chegam, no corpo celular é onde essa informação é processada, axônio transmite a informação processada para os terminais do axônio, esses terminais se conectam a outros dendritos e fazendo uma junção de vários neurônios e essa conexão se chama de sinapse.

Aula 8 - Perceptron de uma camada.

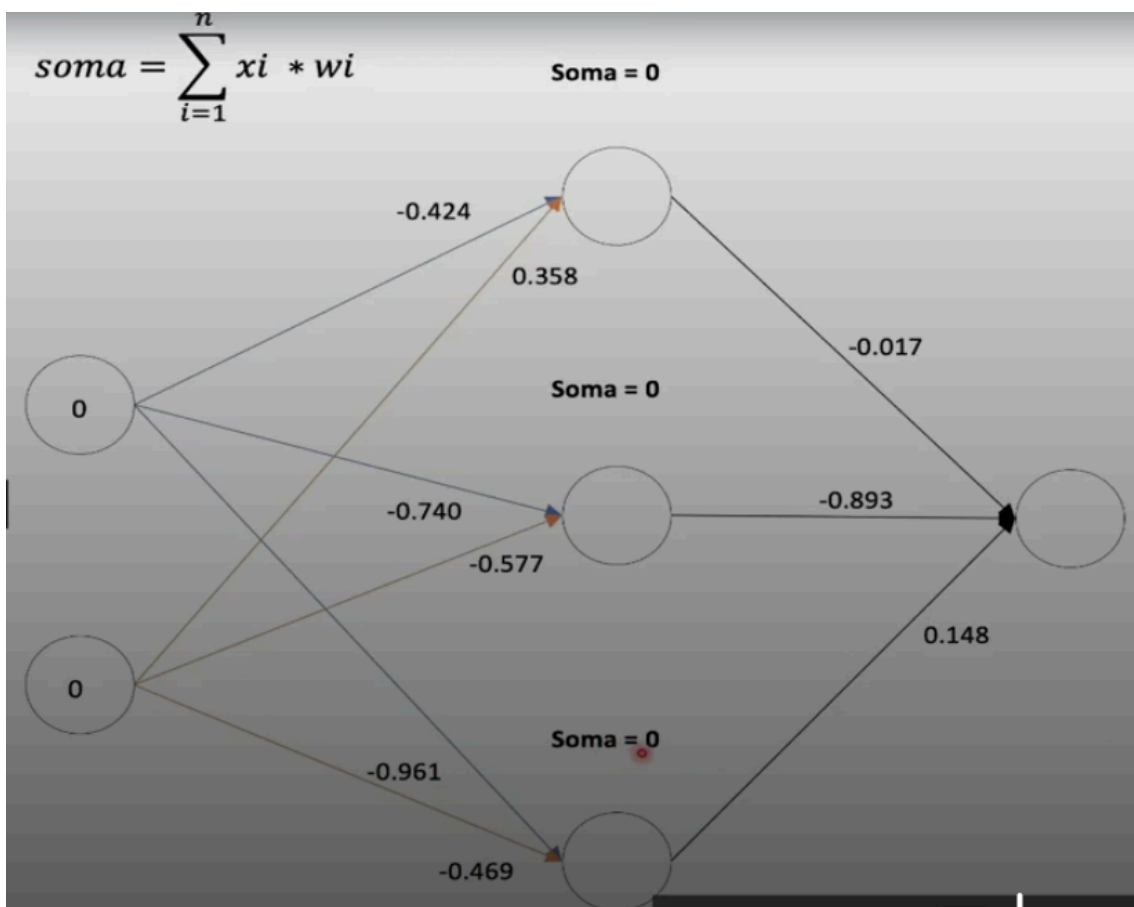
Vai ser nos ensinado como é a representação de um neurônio artificial e de como fazer os cálculos, o neurônio artificial é separado em 3 partes, de começo a entrada, após o peso para cada entrada e depois é a ativação de duas funções, uma função soma e uma função de ativação. É explicado que o objetivo de uma rede neural é encontrar os melhores pesos para as entradas com o intuito de diminuir os erros.

Aula 9 - Redes multicamada - função soma e função de ativação

Nessa aula é utilizado o operador 'Xor' para dar exemplo a rede neural de multicamada, onde o resultado é 1 quando as entradas são diferentes.

x_1	x_2	Saída
1	1	0
1	0	1
0	1	1
0	0	0

No meio da aula ele vai testando caso a caso,



A entrada é 0, é multiplicada pelo peso e passa para a camada oculta onde é aplicado a função de sigmoide para onde nesse caso a função retornará 0,5 para ambas entradas.

Aula 10 - Redes Multicamada - cálculo do erro.

- Algoritmo mais simples

- $\text{erro} = \text{respostaCorreta} - \text{respostaCalculada}$

x1	x2	Classe	Calculado	Erro
0	0	0	0.406	-0.406
0	1	1	0.432	0.568
1	0	1	0.437	0.563
1	1	0	0.458	-0.458

Cálculo feito durante foi esse, do erro ser a resposta certa menos a resposta calculada, a média de erro é a média absoluta, onde o resultado é 0.49, lembrando que o objetivo é sempre ir alterando os pesos fazendo com que esse erro se aproxime cada vez mais de 0.

Aula 11 - Descida do gradiente.

Descida do gradiente é a técnica utilizada para atualizar os pesos com o objetivo de diminuir o erro, basicamente ele encontra a combinação de pesos onde o erro é o menor possível.

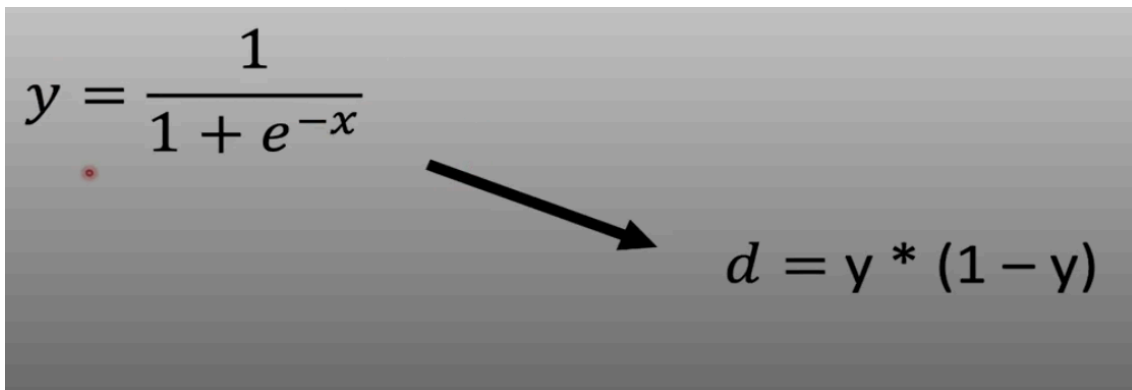
Aula 12 - Cálculo do parâmetro delta.

Esse cálculo do parâmetro delta é útil para saber a direção da atualização dos pesos, segue esse princípio:



Função de ativação, a partir disso a derivada do resultado e assim o delta e só assim para conseguirmos o gradiente dito da aula passada.

A função ativação é a função sigmoide é a derivada que vem do resultado dela, como na imagem abaixo:


$$y = \frac{1}{1 + e^{-x}}$$
$$d = y * (1 - y)$$

Primeira equação é a função de sigmoide e a segunda é a derivada utilizando o resultado obtido, esse cálculo para chegar no final do gradiente serve para conseguirmos chegar no mínimo global da função onde o erro vai ser o menor possível.

O delta é obtido com erro multiplicado com a derivada, é feito esse cálculo para cada registro do caso de estudo que é o XOR, porém esse cálculo é feito para as camadas de saída as camadas ocultas recebem outra fórmula.

O delta da camada oculta é obtido com a derivada multiplicada pelo peso e pelo delta da saída.

Aula 13 - Ajuste dos pesos com backpropagation.

BackPropagation é o algoritmo que é utilizado nas redes neurais ajustando os pesos das conexões, é utilizado esse cálculo em cada camada oculta de cada registro:

$$Peso_{n+1} = (peso_n * momento) + (entrada * delta * taxa de aprendizagem)$$

É feito a soma da entrada * delta para facilitar primeiro, e depois é feito o resto da fórmula, aqui a fórmula aplicada nos pesos e os respectivos resultados:

$$(-0.017 * 1) + 0.032 * 0.3 = -0.007$$

$$(-0.893 * 1) + 0.022 * 0.3 = -0.886$$

$$(0.148 * 1) + 0.021 * 0.3 = 0.154$$

É feito este mesmo processo para a camada de entrada, esse processo todo é feito para atualizar os pesos para a menor quantidade de erros, porém humanamente é complicado e isso passado em códigos com várias épocas é bem mais eficiente.

Aula 14 - Bias, erro, descida do gradiente estocástico e mais parâmetros.

Aqui vai ser mostrado alguns parâmetros adicionais das redes neurais, a primeira é a unidade de bias, é um neurônio especial adicionado a cada camada, para ajustar a ativação de outros neurônios, não é necessário se preocupar com isso pois as bibliotecas já fazem isso transparentemente.

Erro que foi estudado era o algoritmo mais simples, existem formas mais complexas e melhores, MSE e RMSE, mse ou mean squared error é a média quadrada dos erros, tem essa fórmula:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Mean Error Squared

E o RMSE é a raiz quadrada dessa mesma conta, trazendo uma maior penalização dos erros, essa é a fórmula:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^N \|y(i) - \hat{y}(i)\|^2}{N}},$$

Existem duas maneiras de calcular a descida do gradiente, a padrão que já foi nos mostrada em aulas passadas onde é calculado o erro para todos os registros e assim atualizar os pesos, mas a descida do gradiente estocástica funciona de outra maneira, ela calcula o erro para cada registro e atualiza os pesos, a vantagem é que é mais rápido e ajuda a prevenir os mínimos locais dentro da função. E existe o mini batch gradient descent que escolhe um número e registros para rodar e atualizar os pesos.

Aula 24 - K Fold Cross Validation.

K-fold cross-validation é uma técnica de validação onde os dados são divididos em K partes iguais. Em cada iteração, uma parte é usada para teste, enquanto as outras K-1 partes são usadas para treino. O processo é repetido K vezes, garantindo que cada parte seja usada como teste uma vez. No final, uma

média das métricas de avaliação é calculada para medir o desempenho geral do modelo, exemplo dos conjuntos.



Aula 26 - Overfitting e Underfitting.

Overfitting e Underfitting são problemas que acompanham toda essa área de machine learning, redes neurais e entre outras, esses problemas acontecem quando os dados se ajustam de mais ou de menos para modelos e se tornando não tão úteis e eficientes. Overfitting é quando o modelo ou a rede se ajusta demais aos dados incluindo outliers e outros, basicamente quando ele se torna muito próximo apenas aos dados de treinamento. Underfitting é exatamente o contrário quando ele é muito simples e não consegue aprender uma relação concreta, acontece quando os dados de treinamento são limitados. Porém existem correções para esses casos como K-Fold-Cross-Validation é feito para corrigir o Overfitting.

Aula 27 - Overfitting e DropOut Implementação.

A partir dessa aula o conteúdo do curso começa a ficar muito mais prático do que teórico, porém nos é ensinado nessa aula como evitar o overfitting de uma maneira simples utilizando o dropout, no que consiste em 'dropar' uma certa porcentagem que vai definir quantos porcentos dos neurônios não vão ser usados para o treinamento, só é válido em camadas ocultas, já que na camada de entrada fazendo isso se pode perder dados muito importantes.

Aula 28 - Tuning dos parâmetros.

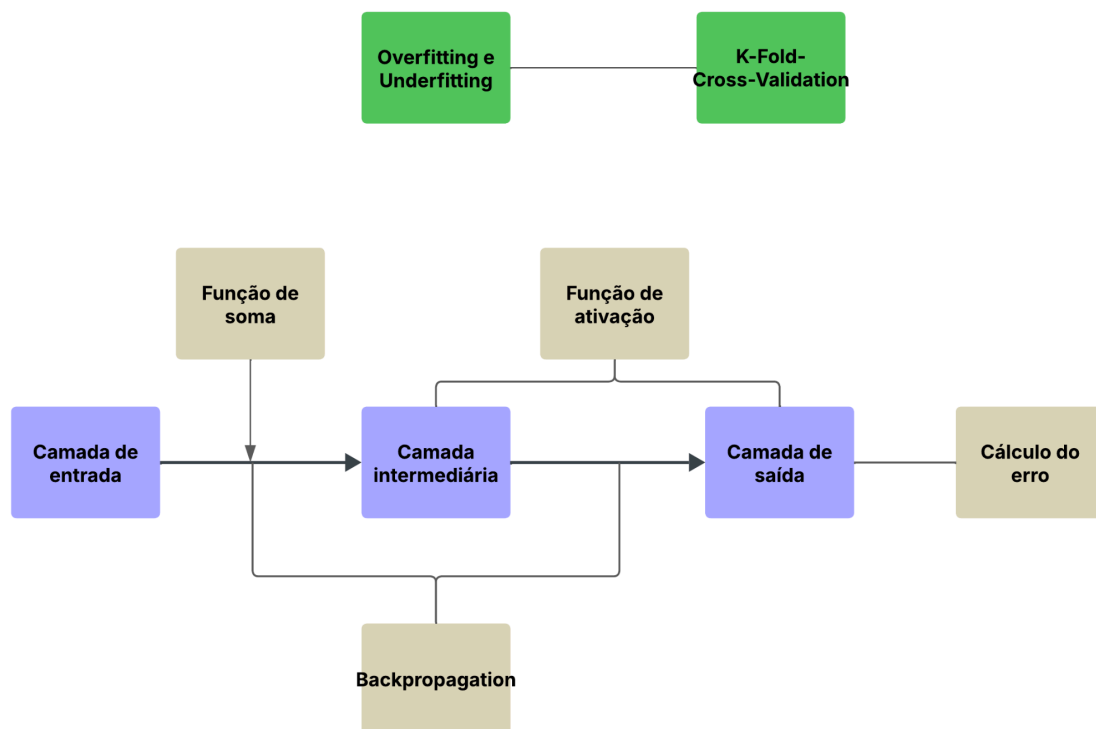
Nessa aula também prática é mostrado como a própria rede neural pode nos mostrar os melhores parâmetros para aquele teste em específico, na aula é

passado 2 tipos diferentes para cada parâmetro e é feito o teste, graças a uma biblioteca do sklearn ele consegue nos retornar quais parâmetros juntos tiveram a melhor taxa de acertos.

Seções 5,6,7:

Essas seções são puramente práticas utilizando bases de dados diferentes, e utilizando todo conhecimento aprendido nas seções anteriores.

Fiz um insight visual original através do LucidChart, trazendo os conceitos mais importantes aprendidos no curso.



Dar uma breve resumida sobre eles apenas para ficar mais claro:

1. **Overfitting e Underfitting:** Overfitting a rede aprende demais os detalhes do treino, inclusive os ruídos, e perde capacidade de generalização. Underfitting a rede não aprende o suficiente, tem baixa performance tanto no treino quanto no teste.
2. **K-Fold-Cross-Validation:** Divide os dados em K partes; treina em $K - 1$ e testa na parte restante, repetindo o processo K vezes. Ajuda a avaliar o desempenho do modelo de forma mais confiável e evitar overfitting.
3. **Camada de entrada:** Recebe os dados brutos e os repassa para a próxima camada sem processamento.
4. **Camada Intermediária:** Realiza o processamento principal usando pesos, soma, ativação e aprende padrões complexos.
5. **Camada Oculta:** Gera a resposta final da rede, como uma classe ou valor, com ativação apropriada para a tarefa.

6. **Função de soma:** Combina os valores de entrada multiplicados por seus respectivos pesos e soma com o bias
7. **Função de ativação:** Aplicada após cada neurônio, transforma a saída linear em uma forma não-linear para aprender padrões complexos.
8. **Backpropagation:** Propaga o erro da saída para trás, atualizando os pesos com base nos deltas para reduzir o erro da rede.
9. **Cálculo de erro:** Compara a saída da rede com o valor real, gerando um valor escalar que representa o quão errada está a predição.

Conclusões

A conclusão que eu chego é que as redes neurais são muito úteis para fazer previsões de dados e que se houver o conhecimento certo para modelar e fazer o pré-processamento dos dados e dos parâmetros necessários pode se tornar uma ferramenta muito poderosa.

Referências