# Relatório <12> — <Prática: Redes Neurais (II)>

<Jonas Correia>

## 1. Fundamentos biológicos

O conteúdo abordado nesta aula introduz os fundamentos biológicos das redes neurais, explicando inicialmente o funcionamento das redes neurais humanas e preparando a compreensão dos conceitos relacionados às redes neurais artificiais. Foi discutida a estrutura dos neurônios e sua interconexão no cérebro, destacando a presença de bilhões de neurônios conectados entre si. Essas conexões, conhecidas como sinapses, são responsáveis pela troca de informações, sendo fundamentais para as habilidades cognitivas e motoras do ser humano, como a fala, visão e locomoção. O processo de aprendizagem, por exemplo, é explicado como resultado da formação de novas conexões neuronais.

Além disso, foi descrita a anatomia básica de um neurônio, enfatizando os dendritos, que recebem os estímulos externos, o corpo celular, onde ocorre o processamento das informações, e o axônio, responsável por transmitir essas informações para outros neurônios. A transmissão de dados entre os neurônios é regulada por impulsos elétricos, que alteram o potencial elétrico do corpo celular durante as sinapses. Esses processos bioelétricos formam a base da comunicação neural, possibilitando o aprendizado de novas habilidades e a adaptação do cérebro humano a diferentes estímulos. A compreensão desses aspectos é fundamental para a transição para o estudo das redes neurais artificiais, tema que será explorado em aulas subsequentes.

## 2. Perceptron de uma camada

A aula abordou os fundamentos do neurônio artificial, explicando sua estrutura básica e o processo de cálculo envolvido no processamento de informações. Um neurônio artificial é composto por três partes principais: as entradas, os pesos associados a essas entradas, e as funções matemáticas de soma e ativação. As entradas representam os dados que o neurônio recebe, e os pesos determinam a importância de cada entrada no cálculo final. A função soma multiplica cada entrada pelo respectivo peso e, em seguida, soma todos os valores. O resultado é passado para a função de ativação, que decide se o neurônio será ativado ou não, com base em um limiar pré-definido.

Para facilitar a compreensão, a Step Function foi utilizada como exemplo de função de ativação. Se o valor da soma for maior ou igual a um, a saída do neurônio é um; caso contrário, é zero. Esse processo foi comparado com o neurônio biológico, onde os dendritos recebem informações, o corpo celular processa esses dados, e os terminais do axônio transmitem o resultado para outros neurônios.

O conceito de perceptron de uma camada foi aplicado a um estudo de caso envolvendo o operador lógico E. Nessa aplicação, duas entradas (x1 e x2) são processadas para prever a classe de saída. O objetivo da rede neural é encontrar os melhores pesos para classificar corretamente as entradas. Ao longo da explicação, demonstrou-se o processo de cálculo da função soma, a aplicação da função de ativação e a atualização dos pesos com base no erro da previsão inicial.

Além disso, foi discutida a linearidade de certos operadores lógicos, como o E e o OU, que são linearmente separáveis, permitindo que uma linha reta divida suas classes de forma correta. No entanto, o operador XOR foi apresentado como um problema não linearmente separável, o que exige uma rede neural mais complexa, como as redes neurais multicamadas. Essa limitação do perceptron de uma camada leva à necessidade de estruturas mais sofisticadas, que serão exploradas em aulas futuras.

## 3. Redes multicamada - função soma e função de ativação

Na aula, foi explorada a teoria das redes neurais multicamadas, também conhecidas como Multi-Layer Perceptron (MLP). Inicialmente, foi feita uma revisão sobre o perceptron de uma única camada, destacando suas limitações para resolver problemas que não são linearmente separáveis. A partir disso, a solução apresentada foi a adição de múltiplas camadas de perceptron, formando uma rede neural multicamada. Cada neurônio da camada de entrada é conectado aos neurônios de uma camada oculta, que, por sua vez, estão conectados ao neurônio da camada de saída. Essa estrutura é chamada de rede neural densa, onde todos os neurônios de uma camada se conectam a todos os neurônios da próxima.

O processo de cálculo para uma rede neural multicamada mantém a essência do perceptron de camada única, com a adição de uma função soma e uma função de ativação para cada neurônio. A função de ativação discutida foi a função sigmoide, amplamente utilizada em problemas mais complexos devido à sua capacidade de retornar valores entre 0 e 1, o que a torna útil para classificações probabilísticas.

O exemplo prático abordado foi o operador lógico XOR, que, sendo um problema não linearmente separável, não pode ser resolvido por uma rede neural de camada única. Assim, foi mostrado o processo de definição de pesos aleatórios e a aplicação das funções soma e sigmoide para uma rede multicamada. As previsões da rede neural foram comparadas com as respostas esperadas, revelando erros que indicam a necessidade de ajustes nos pesos da rede.

## 4. Redes multicamada - cálculo do erro

Na aula foi apresentado o cálculo do erro em uma rede neural, partindo da ideia de comparação entre os valores previstos pela rede e os valores reais. Inicialmente, foi recordado o processo de passagem de dados pela rede, desde a

aplicação da função soma e da função de ativação, tanto na camada oculta quanto na camada de saída. A partir disso, foram apresentados os valores de previsão gerados pela rede para cada registro de entrada, e o próximo passo foi calcular o erro comparando esses valores com os dados reais da base de treinamento.

A fórmula mais simples para calcular o erro consiste na subtração entre a resposta correta e a previsão da rede. No entanto, como o objetivo é observar a magnitude do erro, foi utilizado o valor absoluto, ignorando qualquer valor negativo. Assim, a média absoluta do erro foi calculada somando os valores individuais e dividindo pelo número de registros, resultando em um erro médio de 0,49.

O objetivo principal é reduzir esse valor de erro nas próximas iterações. Isso será feito através de um algoritmo de atualização de pesos, que ajustará os pesos da rede neural para melhor se adaptarem aos dados fornecidos. Em cada rodada, ou época, os pesos são recalculados, e a expectativa é que o erro vá diminuindo progressivamente. O número de épocas, que pode ser configurado pelo usuário, será um fator crucial para determinar o grau de refinamento da rede neural.

## 5. Cálculo do parâmetro delta

O cálculo do parâmetro Delta é um passo essencial no processo de atualização dos pesos em redes neurais, sendo fundamental para determinar a direção correta dessa atualização antes da aplicação do gradiente. Esse parâmetro é obtido por meio do cálculo do erro multiplicado pela derivada da função de ativação, que utiliza a função sigmoide. O Delta é calculado tanto para a camada de saída quanto para a camada oculta da rede.

Na camada de saída, o erro é a diferença entre o valor real e o valor previsto pela rede. Para calcular o Delta, utiliza-se a derivada da função sigmoide, que reflete a taxa de variação da função de ativação. Esse processo é repetido para cada registro da base de dados, garantindo que cada peso seja atualizado de maneira adequada para reduzir o erro.

Após realizar o cálculo do Delta para a camada de saída, é necessário proceder com o mesmo cálculo para a camada oculta. Nesse caso, o valor da ativação da camada oculta é utilizado para calcular a derivada da função sigmoide. O Delta da camada oculta é então obtido multiplicando-se o valor da derivada pelos pesos e pelo Delta da camada de saída. Esse cálculo permite ajustar os pesos da camada oculta para que a rede neural possa minimizar o erro de maneira mais eficaz.

Com o Delta calculado tanto para a camada de saída quanto para a camada oculta, o próximo passo será a atualização dos pesos, que ocorre com base nesses valores para garantir que o erro da rede neural continue a diminuir ao longo das iterações.

## 6. Ajuste dos pesos com backpropagation

O processo de ajuste dos pesos em uma rede neural é fundamental para o aprendizado e a convergência aos pesos ideais. Na abordagem de retro propagação, conhecida como backpropagation, primeiramente realizam-se cálculos da camada de entrada até a camada de saída, passando pela função soma e ativação. Após isso, a direção do cálculo é invertida, voltando da camada de saída até a de entrada. Isso permite ajustar os pesos com base no erro da rede.

A fórmula utilizada para a atualização dos pesos envolve a multiplicação da entrada pelo delta, somada ao valor do peso anterior. Esse processo inclui um parâmetro chamado de "momento", que acelera a convergência, e a "taxa de aprendizagem", configurada pelo usuário. Esta última controla a velocidade de ajuste dos pesos, podendo ser mais alta ou baixa, influenciando diretamente o tempo necessário para a rede atingir o mínimo global, onde o erro é minimizado.

Os cálculos do delta nas camadas de saída e oculta são usados para determinar a direção da atualização dos pesos. Para a camada oculta, o valor do delta é calculado multiplicando-se a derivada da função de ativação pelos pesos anteriores e pelo delta da camada de saída. Já os pesos são ajustados considerando a entrada multiplicada pelo delta e a taxa de aprendizagem. Esse processo é repetido por várias épocas, e a cada repetição, a rede neural se aproxima de encontrar o melhor conjunto de pesos para classificar os registros corretamente.

Portanto, o ajuste de pesos, guiado pelo algoritmo de retro propagação, é o que permite a aprendizagem da rede neural, garantindo que ela possa melhorar seu desempenho ao longo do tempo, reduzindo os erros e se aproximando de soluções otimizadas.

## 7. Bias, erro, descida do gradiente estocástico e mais parâmetros

As redes neurais possuem parâmetros adicionais que influenciam diretamente no processo de aprendizado. Um desses elementos é a unidade de bias, que, embora adicione um valor fixo, tem pesos que também são ajustados durante o treinamento, permitindo que a rede produza saídas não nulas, mesmo quando todas as entradas são zero. Isso melhora o desempenho geral, sendo implementado automaticamente em muitas bibliotecas de aprendizado de máquina.

Outro aspecto fundamental é a forma de calcular o erro. O erro simples, derivado da diferença entre a resposta correta e a resposta calculada, pode ser insuficiente em problemas complexos. Para isso, métricas como o Mean Squared Error (MSE) e o Root Mean Squared Error (RMSE) são amplamente utilizadas.

Essas métricas penalizam mais os erros maiores, facilitando a detecção e correção de desvios significativos no treinamento.

Na estrutura de redes neurais para classificação, a codificação de saídas em múltiplos neurônios reflete a multiplicidade de classes, como no exemplo de prever o risco de crédito. A quantidade de neurônios na camada de saída depende do número de classes. Para classificar três níveis de risco, por exemplo, são usados três neurônios, com saídas codificadas para cada classe.

No treinamento de redes, o Batch Gradient Descent calcula o erro para todos os registros antes de atualizar os pesos. Já o Stochastic Gradient Descent calcula o erro e atualiza os pesos a cada registro, prevenindo a convergência em mínimos locais. O Mini-Batch Gradient Descent combina os dois métodos, atualizando os pesos após processar lotes de registros, sendo o mais comum em aplicações práticas.

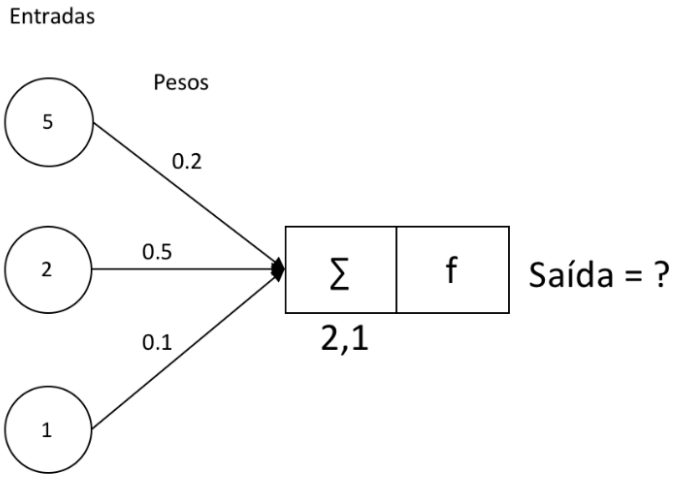
Outros parâmetros, como a taxa de aprendizagem (learning rate), o tamanho do lote (batch size) e as épocas (epochs), determinam a eficiência e a precisão do modelo. A taxa de aprendizagem influencia a rapidez com que os pesos são ajustados, enquanto o batch size e o número de épocas controlam o processo de atualização dos pesos ao longo do treinamento.

Finalmente, diversas funções de ativação, como a sigmoid, tangente hiperbólica, ReLU e softmax, desempenham papéis críticos na transformação das saídas dos neurônios. Funções como a softmax, por exemplo, são essenciais para classificação multiclasse, retornando probabilidades para cada classe possível. Em problemas de regressão, a função linear é amplamente usada por permitir a previsão de valores contínuos, como limites de crédito, demonstrando a flexibilidade das redes neurais em diferentes cenários de aplicação.

# 8. Teste 1: Teoria sobre redes neurais artificiais

Pergunta 2:

Considerando o resultado da aplicação da função soma = 2,1 para o perceptron abaixo, quais são os valores da aplicação das funções `sigmoide` , `tangente hiperbólica` , `relu` e `linear` ; respectivamente?



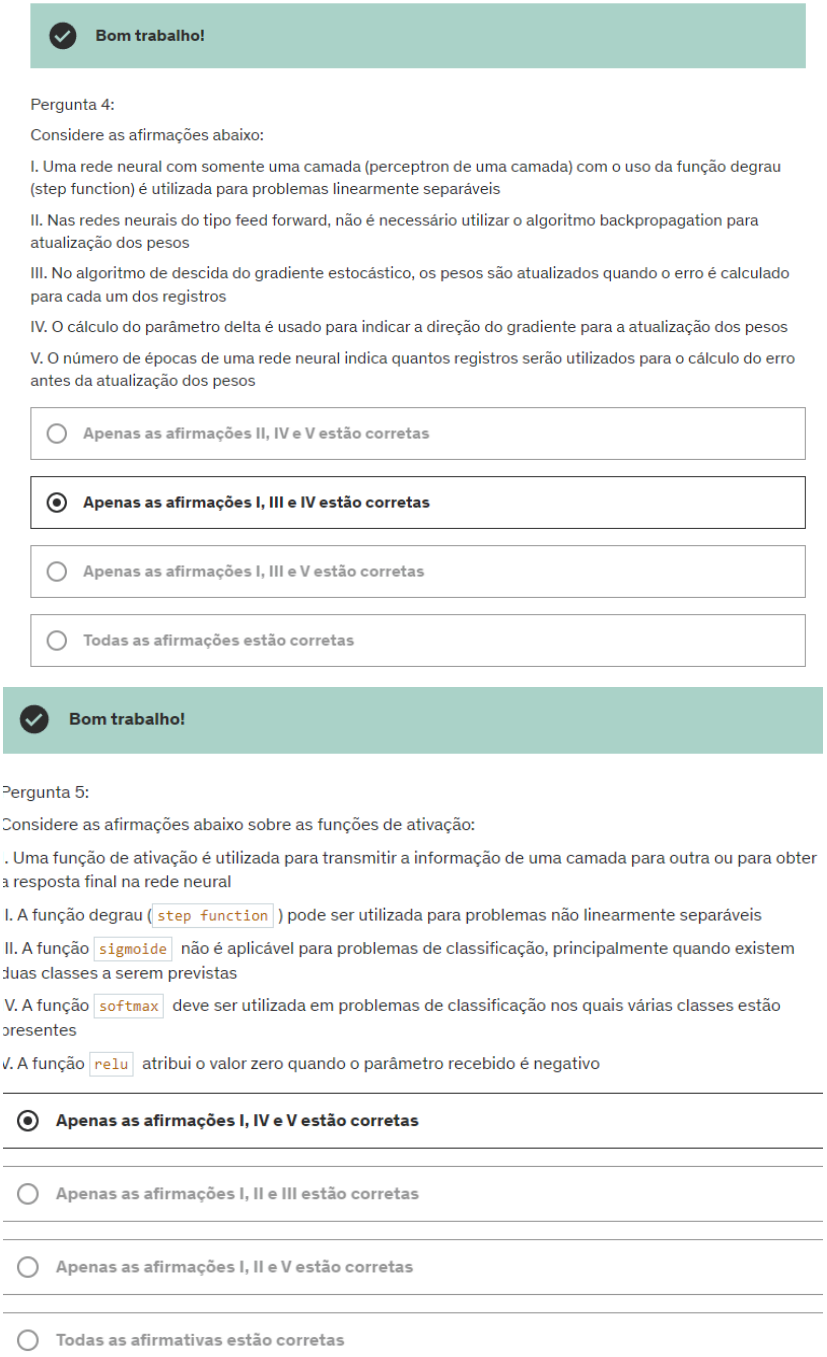

- ○ 0,89 0,97 2,1 1,2
- ○ 0,89 0,90 2,1 2,1
- ◉ **0,89 0,97 2,1 2,1**
- ○ 0,98 0,97 1,2 2,1

Pergunta 3:

A tabela abaixo apresenta na primeira coluna a classe esperada e na segunda coluna os resultados que foram calculados pela rede neural. Com base nesses dados, indique os valores do `mean absolute error` , `mean squared error` , `root mean squared error` e a `taxa de acerto (%)` da rede neural; respectivamente

| Classe | Calculado |
|---|---|
| 1 | 0.300 |
| 0 | 0.020 |
| 1 | 0.890 |
| 0 | 0.320 |

- ○ 0,28 0,10 0,45 e 90%
- ○ 0,28 0,40 0,16 e 28%
- ○ 0,16 0,40 0,28 e 54%
- ◉ **0,28 0,15 0,38 e 76,5%**




# 9. Validação cruzada - teoria

A validação cruzada é uma técnica essencial em aprendizado de máquina para avaliar a desempenho de um modelo de forma mais robusta e confiável. Em vez de simplesmente dividir a base de dados em dois conjuntos, um para treinamento e outro para teste, a validação cruzada permite que todos os registros da base de dados sejam utilizados tanto para treinamento quanto para teste, aumentando a eficácia da avaliação.

A abordagem comum na validação cruzada, conhecida como K-fold cross-validation, envolve dividir a base de dados em K subconjuntos ou "folds". No exemplo mencionado, foram utilizados 4 folds, mas, na prática, o número 10 é frequentemente empregado devido à sua aceitação em estudos científicos. Cada um desses folds contém uma parte dos dados que não foi utilizada para treinar o modelo em uma iteração específica.

Durante o processo, o modelo é treinado em K-1 folds e testado no fold restante. Isso é repetido K vezes, com cada fold sendo utilizado uma vez como conjunto de teste. Essa abordagem não apenas garante que todos os dados sejam utilizados, mas também ajuda a evitar a possibilidade de que informações importantes sejam excluídas da fase de treinamento.

Ao final do K-fold cross-validation, você terá K medidas de desempenho, como a acurácia (accuracy) do modelo. Para obter uma avaliação final do modelo, calcula-se a média dessas medidas. Por exemplo, se os percentuais de acerto foram de 85%, 79%, 84% e 77%, a média seria de 81,25%. Essa média fornece uma estimativa mais precisa da capacidade do modelo em generalizar para novos dados.

A validação cruzada não só melhora a confiança nas estimativas de desempenho do modelo, mas também é uma prática recomendada em situações em que se deseja minimizar a variabilidade na avaliação do modelo. Na próxima aula, a comparação entre a validação cruzada e a divisão simples entre treinamento e teste será uma tarefa interessante, pois ajudará a solidificar o entendimento sobre a importância dessa técnica na prática de aprendizado de máquina.

## 10. Overfitting e underfitting - teoria

Nesta aula, abordamos dois problemas fundamentais em aprendizagem de máquina e redes neurais: o overfitting e o underfitting, que refletem a qualidade da adaptação de um modelo aos dados de treinamento e sua capacidade de generalizar para novos dados.

O underfitting ocorre quando o modelo é excessivamente simples para capturar a complexidade do problema. Em termos práticos, isso significa que, ao testar o modelo, ele apresenta uma precisão muito baixa, tanto nos dados de treinamento quanto nos de teste. É como tentar resolver um problema complexo (representado pela figura de um tiranossauro) usando um método simplório, como uma raquete de mosquitos. No gráfico, um modelo em underfitting pode ser visualizado como uma linha reta que passa longe de muitos pontos de dados, indicando que o modelo não consegue capturar as variações necessárias para classificar os dados de maneira eficaz.

Por outro lado, o overfitting acontece quando o modelo é excessivamente complexo, adaptando-se demais aos dados de treinamento e, por isso, não é capaz de generalizar para dados novos. Visualmente, o modelo segue uma linha que toca quase todos os pontos de dados, como se tivesse "decorado" os padrões específicos do conjunto de treinamento. Embora o overfitting leve a uma alta precisão nos dados de treinamento, os resultados caem drasticamente ao testar o modelo com novos dados, indicando que ele apenas memorizou o conjunto de treinamento em vez de aprender padrões gerais.

Para evitar esses problemas, é essencial adequar a complexidade do modelo à complexidade do problema. Se o problema é simples, um modelo mais básico, como uma regressão linear, pode ser suficiente; já para problemas complexos, redes neurais com camadas adicionais podem ser mais apropriadas. Para identificar a ocorrência de underfitting, observa-se uma baixa acurácia nos dados de treinamento e de teste. Já o overfitting é detectado quando a precisão no treinamento é alta, mas cai consideravelmente nos testes.

Essa compreensão é vital para escolhermos o modelo certo para cada problema, garantindo um bom desempenho tanto nos dados de treinamento quanto em novos dados. Em suma, evitar o underfitting e o overfitting nos aproxima de um modelo que consegue equilibrar adaptação e generalização, promovendo um aprendizado eficaz.