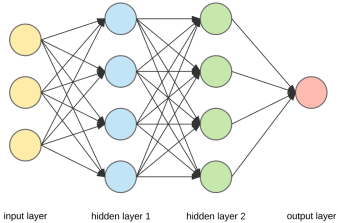
Parte IV: Aprendizado de Máquina.

Prof. Fabiano Araujo Soares, Dr. / FGA 0221 - Inteligência Artificial

Universidade de Brasília

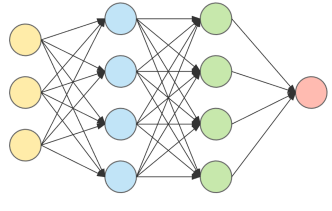
2025

Aprendizado Supervisionado: Redes Neurais Artificiais



Se nosso cérebro é capaz de aprender e tomar decisões complexas a partir de informações que recebe, como vocês imaginam que uma máquina pode aprender a reconhecer padrões ou tomar decisões semelhantes?

Aprendizado Supervisionado: Redes Neurais Artificiais



Se nosso cérebro é capaz de aprender e tomar decisões complexas a partir de informações que recebe, como vocês imaginam que uma máquina pode aprender a reconhecer padrões ou tomar decisões semelhantes?

Redes neurais artificiais funcionam inspiradas no cérebro humano, compostas por inúmeras unidades que processam e combinam informações de forma a aprenderem padrões complexos.

Por que estudar Redes Neurais?

- Redes neurais têm a capacidade de aprender com grandes volumes de dados, identificando padrões complexos e não lineares que métodos tradicionais não conseguem capturar.
- Elas são aplicadas em setores cruciais como **saúde** (diagnóstico por imagens, assistência médica), **finanças** (detecção de fraudes, previsões de mercado), **indústria** (controle de qualidade, otimização de processos) e **entretenimento** (recomendações personalizadas).
- Ao aprenderem a generalizar a partir de exemplos, redes neurais permitem que computadores tomem decisões inteligentes com pouca intervenção humana, resolvendo problemas complexos e adaptando-se a cenários dinâmicos.
- Estudar redes neurais é essencial para criar soluções inovadoras em inteligência artificial, que estão transformando o modo como vivemos e trabalhamos.

Inspiração Biológica

- Baseadas no funcionamento dos neurônios do cérebro humano.
- Unidades simples (neurônios artificiais) conectadas formando redes.
- Capacidade coletiva de processar e aprender informações.

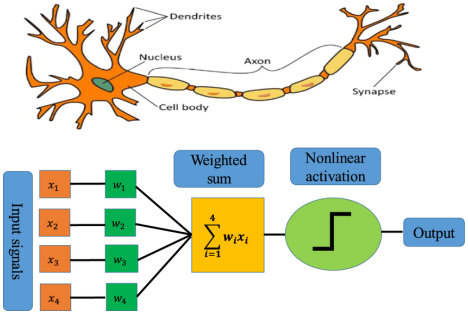


Figura: Neurônio biológico vs. neurônio artificial.

O que é uma Rede Neural?

- **Uma rede neural artificial - ANN** é um modelo computacional inspirado na estrutura do cérebro humano, formado por unidades chamadas **neurônios artificiais**.
- Esses neurônios são organizados em camadas: a **camada de entrada** recebe os dados; **camadas ocultas** processam informações em vários níveis de abstração; e a **camada de saída** entrega o resultado final.
- Cada conexão entre neurônios possui um **peso** que determina a influência do sinal transmitido, e cada neurônio aplica uma **função de ativação** para gerar sua saída.
- Redes neurais podem resolver tarefas como **classificação**, **regressão**, **previsão** e **reconhecimento**, aprendendo a partir de exemplos e ajustando seus pesos de forma automática.

Exemplos Reais de Aplicação

- **Diagnóstico médico por imagem**, como análise de radiografias e ressonâncias magnéticas, que auxiliam na detecção precoce de doenças e no monitoramento de tratamentos.
- **Detecção de fraudes em transações financeiras**, utilizando padrões de comportamento para identificar atividades suspeitas e proteger consumidores e instituições.
- **Sistemas de recomendação em plataformas de streaming e e-commerce**, que personalizam conteúdos e produtos com base no histórico e preferências dos usuários.
- **Tradução automática e assistentes virtuais**, que interpretam e geram linguagem natural para facilitar a comunicação e automação de tarefas diárias.
- **Carros autônomos e visão computacional**, permitindo que veículos analisem o ambiente, tomem decisões em tempo real e naveguem com segurança.

Importância das Redes Neurais

- Redes neurais geram uma função interna que mapeia a relação entre entradas e saídas dos dados de treinamento.
- Permitem que máquinas aprendam automaticamente a partir de grandes volumes de dados.
- Elas automatizam processos, aumentando a precisão das operações e otimizando a tomada de decisões em ambientes dinâmicos.
- A versatilidade das redes neurais viabiliza aplicações diversas, desde diagnósticos médicos até sistemas financeiros e industriais, adaptando-se a múltiplos contextos.
- Esse conjunto de características torna o estudo das redes neurais fundamental para a inovação tecnológica e a transformação digital em vários setores.

Limitações das Redes Neurais

- Requerem **grandes volumes de dados** para treinamento eficaz, o que pode ser caro ou inviável em alguns contextos.
- **Alto custo computacional**, especialmente em redes profundas e modelos complexos.
- **Risco de overfitting**, quando o modelo se ajusta demais aos dados de treinamento e perde capacidade de generalização.
- **Caixa-preta**: dificuldade de interpretar e explicar decisões internas do modelo.

Cuidados e Boas Práticas

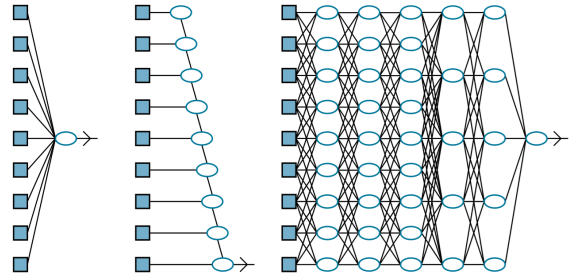
- Garantir qualidade e **representatividade** dos dados para evitar **viés** e garantir **generalização**.
- Utilizar técnicas como **regularização**, **validação cruzada** e ***early stopping*** para prevenir overfitting.
- Avaliar conscientemente a **interpretabilidade** versus **complexidade do modelo**, especialmente em aplicações críticas.
- Considerar o **impacto ético** e a **transparência** no uso de redes neurais em decisões automatizadas.

Considerações Finais

- Redes neurais são poderosas, porém não são solução universal para todos os problemas.
- Entender suas limitações e cuidar da qualidade dos dados são passos críticos para obter resultados confiáveis.
- O desenvolvimento ético e responsável acompanha o uso dessas tecnologias, visando benefícios sociais e minimização de riscos.

Funcionamento das Redes Neurais Artificiais

- A ideia básica das redes neurais (principalmente as *deep learning*) é treinar circuitos de modo que os caminhos de cálculos sejam longos, permitindo que todas as variáveis de entrada interajam de maneiras complexas.



(a) linear regression (b) decision list network (c) A deep learning network

Funcionamento das Redes Neurais Artificiais

- Esses modelos de circuitos se mostram suficientemente expressivos para capturar a complexidade dos dados do mundo real para muitos tipos importantes de problemas de aprendizagem;
- Exemplos de algoritmos de redes neurais são:
  - Redes neurais multi-camadas – MLPs (ou *Feedforward Networks*);
  - Redes neurais convolucionais;
  - Redes neurais recorrentes (RNN);

Feedforward network

- Uma rede *feedforward*, tem conexões apenas em uma direção – ou seja, ela forma um grafo acíclico direcionado com nós de entrada e saída designados;
- Cada nó calcula uma função de suas entradas e passa o resultado para seus sucessores na rede;
- Informação flui através da rede a partir dos nós de entrada em direção aos nós de saída, e não há loops.
- Circuitos que implementam funções booleanas, são um exemplo de redes feedforward. Em um circuito booleano, as entradas são limitadas a 0 e 1, e cada nó implementa um função booleana simples de suas entradas, produzindo um 0 ou um 1 na saída.

Feedforward network

- Algumas das entradas para os nós são parâmetros da rede; a rede aprende ajustando os valores desses parâmetros para que a rede como um todo se ajuste aos dados de treinamento.
- Cada nó dentro de uma rede é chamado de **unidade (McCulloch e Pitts)**, uma unidade calcula a soma ponderada das entradas dos nós predecessores e, em seguida, aplica uma função não linear para produzir sua saída:

$$a_j = g_j(\sum_i \omega_{i,j} a_i) \equiv g_j(in_i)$$

onde a_j é a saída da unidade j, $\omega_{i,j}$ é o peso da ligação entre a unidade i e a unidade j e g_j é a função de ativação não linear associada a unidade j.

Feedforward network

- Cada unidade tem uma entrada extra "0" que é fixada em +1 (**bias**) e um peso $w_{0,j}$ para essa entrada.
- Isso permite que a entrada ponderada total in_j para unidade j seja diferente de 0 mesmo que as saídas da camada anterior sejam todas 0.
- Com essa convenção, podemos escrever a equação anterior na forma vetorial:

$$a_j = g_i(\boldsymbol{\omega}^\top \boldsymbol{X})$$

onde $\boldsymbol{\omega}$ é o vetor de pesos que leva à unidade j (incluindo $\omega_{0,j}$) e $\boldsymbol{X}$ é o vetor de entradas para a unidade j (incluindo o +1).

Feedforward network

- O fato de a **função de ativação** ser **não linear** é importante pois, caso contrário, qualquer composição de unidades ainda representaria uma função linear;
- A não linearidade é o que permite que redes suficientemente grandes de unidades representem funções arbitrárias;
- O **teorema universal da aproximação** afirma que uma rede com apenas duas camadas de unidades de cálculo, a primeira não linear e a segunda linear, pode aproximar qualquer função contínua a um valor arbitrário de precisão.

Feedforward network

- Existe uma grande variedade de **funções de ativação**, as mais comuns delas são:
  - Função logística ou sigmoid:
$$\sigma(x) = \frac{1}{(1 + e^{-x})}$$
  - A função ReLU (rectified linear unit):
$$ReLU(x) = max(0, x)$$
  - A função softplus:
$$softplus(x) = log(1 + e^{x})$$
  - A função tanh:
$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

Feedforward network

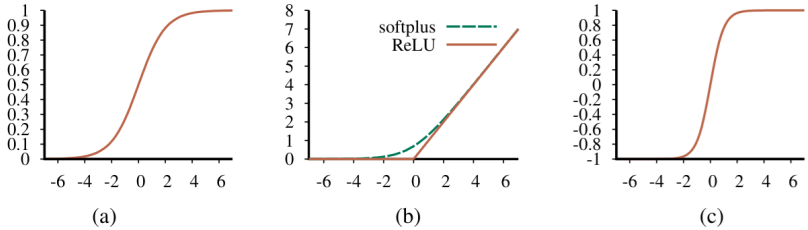


Figure 21.2 Activation functions commonly used in deep learning systems: (a) the logistic or sigmoid function; (b) the ReLU function and the softplus function; (c) the tanh function.

Feedforward network

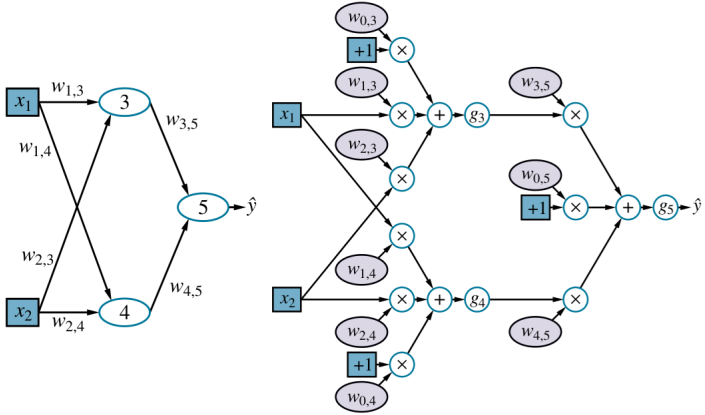


Figure 21.3 (a) A neural network with two inputs, one hidden layer of two units, and one output unit. Not shown are the dummy inputs and their associated weights. (b) The network in (a) unpacked into its full computation graph.

Feedforward network

Para computar a saída do exemplo anterior, fazemos

$$\begin{aligned}\hat{y} &= g_5(in_5) = g_5(\omega_{0,5} + \omega_{3,5}a_3 + \omega_{4,5}a_4) \\ &= g_5(\omega_{0,5} + \omega_{3,5}g_3(in_3) + \omega_{4,5}g_4 in_4) \\ &= g_5(\omega_{0,5} + \omega_{3,5}g_3(\omega_{0,3} + \omega_{1,3}x_1 + \omega_{2,3}x_2) + \omega_{4,5}g_4(\omega_{0,4} + \omega_{1,4}x_1 + \omega_{2,4}x_2)).\end{aligned}$$

Algoritmos de treinamento

- O treinamento de uma rede neural consiste em modificar os parâmetros da rede de forma a minimizar a função de perda (*loss function*) no conjunto de treinamento;
- A função de perda quantifica a diferença entre o resultado esperado e o resultado produzido pelo modelo de aprendizado de máquina.
- Em princípio, qualquer tipo de algoritmo de otimização poderia ser usado;
- Na prática, as redes neurais modernas quase sempre são treinadas com alguma variante de **gradiente descendente estocástico** (*stochastic gradient descent* - SGD).

Algoritmos de treinamento

- Em linhas gerais, a ideia é minimizar a função de perda $L(\boldsymbol{\omega})$, onde $\boldsymbol{\omega}$ representa todos os parâmetros da rede;
- Cada passo no processo SGD é da seguinte forma:

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \alpha \nabla_{\boldsymbol{\omega}} L(\boldsymbol{\omega})$$

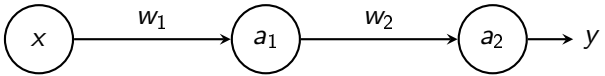
  Onde α é a taxa de aprendizado.
- Para gradiente descendente padrão, a perda $\boldsymbol{L}$ é definida em relação a todo o conjunto de treinamento.
- Para **SGD**, é definido em relação a um subconjunto de m exemplos escolhidos aleatoriamente em cada etapa.

Algoritmo de Backpropagation

Exemplo em uma rede simples de 3 camadas

Rede Neural Simples (3 camadas)

- Rede neural com:
  - 1 neurônio na camada de entrada (entrada x).
  - 1 neurônio na camada oculta (ativação a_1).
  - 1 neurônio na camada de saída (ativação a_2).
- Pesos:
  - w_1: peso entre a entrada x e a camada oculta.
  - w_2: peso entre a camada oculta e a camada de saída.
- Saída desejada (rótulo): y.



Propagação Direta (Forward Pass)

- Entrada escalar: x.
- Camada oculta:

$$z_1 = w_1 x$$

$$a_1 = f(z_1)$$

- Camada de saída:

$$z_2 = w_2 a_1$$

$$a_2 = f(z_2)$$

- Função de ativação $f(\cdot)$ pode ser, por exemplo, linear $f(u) = u$ (função identidade) ou não-linear (sigmoide, ReLU, etc.).

Função de Custo

- Saída da rede: a_2.
- Saída desejada (rótulo): y.
- Usaremos o erro quadrático médio (para um único exemplo):

$$C = \frac{1}{2}(a_2 - y)^2$$

- Objetivo do treinamento:
  - Ajustar w_1 e w_2 para minimizar C.

Ideia do Backpropagation

- Backpropagation calcula como o erro C varia em relação aos pesos.
- Queremos os gradientes:

$$\frac{\partial C}{\partial w_2} \quad \text{e} \quad \frac{\partial C}{\partial w_1}$$

- Usamos a regra da cadeia para decompor essas derivadas em termos mais simples (derivadas locais).

$$\frac{d}{dx} f(g(x)) = f'(g(x))g'(x) = \frac{df}{dg}\frac{dg}{dx}$$

- Em seguida, atualizamos os pesos com gradiente descendente:

$$w \leftarrow w - \eta \frac{\partial C}{\partial w}$$

onde η é a taxa de aprendizado.

Cálculo de $\partial C/\partial w_2$

- Lembrando:

$$C = \frac{1}{2}(a_2 - y)^2, \quad a_2 = f(z_2), \quad z_2 = w_2 a_1$$

- Pela regra da cadeia:

$$\frac{\partial C}{\partial w_2} = \frac{\partial C}{\partial a_2}\frac{\partial a_2}{\partial z_2}\frac{\partial z_2}{\partial w_2}$$

- Termo a termo:

$$\frac{\partial C}{\partial a_2} = a_2 - y$$
$$\frac{\partial a_2}{\partial z_2} = f'(z_2)$$
$$\frac{\partial z_2}{\partial w_2} = a_1$$

- Logo:

$$\frac{\partial C}{\partial w_2} = (a_2 - y)\, f'(z_2)\, a_1$$

Cálculo de $\partial C/\partial w_1$ — Parte 1

- Agora w_1 afeta C através de a_1 e depois a_2:

$$w_1 \to z_1 \to a_1 \to z_2 \to a_2 \to C$$

- Pela regra da cadeia:

$$\frac{\partial C}{\partial w_1} = \frac{\partial C}{\partial a_2}\frac{\partial a_2}{\partial z_2}\frac{\partial z_2}{\partial a_1}\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_1}$$

- Termos iniciais:

$$\frac{\partial C}{\partial a_2} = a_2 - y$$

$$\frac{\partial a_2}{\partial z_2} = f'(z_2)$$

Cálculo de $\partial C/\partial w_1$ — Parte 2

- Continuando os termos:

$$\frac{\partial z_2}{\partial a_1} = w_2$$

$$\frac{\partial a_1}{\partial z_1} = f'(z_1)$$

$$\frac{\partial z_1}{\partial w_1} = x$$

- Portanto a derivada completa é:

$$\frac{\partial C}{\partial w_1} = (a_2 - y)\, f'(z_2)\, w_2\, f'(z_1)\, x$$

Atualização dos Pesos (Gradiente Descendente)

- Dada a taxa de aprendizado $\eta > 0$, atualizamos:

$$w_2 \leftarrow w_2 - \eta \frac{\partial C}{\partial w_2}$$

$$w_1 \leftarrow w_1 - \eta \frac{\partial C}{\partial w_1}$$

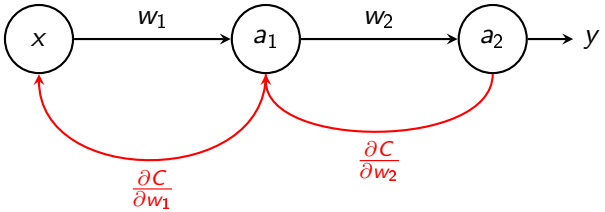
- Em forma explícita:

$$w_2 \leftarrow w_2 - \eta \left(a_2 - y\right) f'\left(z_2\right) a_1$$

$$w_1 \leftarrow w_1 - \eta \left(a_2 - y\right) f'\left(z_2\right) w_2 \, f'\left(z_1\right) x$$

- Repetimos esse processo para vários exemplos de treinamento e várias épocas, fazendo a rede aprender a mapear $x \mapsto y$.

Esquemático do Backpropagation



- As setas pretas representam o fluxo de informação (forward).
- As setas vermelhas representam o fluxo do erro (gradientes) voltando pela rede (backpropagation).

Exemplo Numérico Simples

- Considere:

$$x = 1.5, \quad y = 0.5, \quad w_1 = 0.8, \quad w_2 = 0.5, \quad f(u) = u$$

- Forward:

$$a_1 = w_1 x = 0.8 \cdot 1.5 = 1.2$$

$$a_2 = w_2 a_1 = 0.5 \cdot 1.2 = 0.6$$

- Custo:

$$C = \frac{1}{2}(a_2 - y)^2 = \frac{1}{2}(0.6 - 0.5)^2$$

Exemplo Numérico: Cálculo dos Gradientes

- Dados anteriores: $x = 1.5,\ y = 0.5,\ w_1 = 0.8,\ w_2 = 0.5,\ f(u) = u$.
- Função identidade tem derivada: $f'(u) = 1$.
- Cálculo das derivadas parciais:

$$\frac{\partial C}{\partial w_2} = (a_2 - y) \cdot f'(z_2) \cdot a_1 = (0.6 - 0.5) \cdot 1 \cdot 1.2 = 0.12$$

$$\frac{\partial C}{\partial w_1} = (a_2 - y) \cdot f'(z_2) \cdot w_2 \cdot f'(z_1) \cdot x = (0.6 - 0.5) \cdot 1 \cdot 0.5 \cdot 1 \cdot 1.5 = 0.075$$

Exemplo Numérico: Ajuste dos Pesos

- Taxa de aprendizado escolhida: $\eta = 0.1$.
- Atualização dos pesos via gradiente descendente:

$$w_2^{\text{novo}} = w_2 - \eta \frac{\partial C}{\partial w_2} = 0.5 - 0.1 \times 0.12 = 0.488$$

$$w_1^{\text{novo}} = w_1 - \eta \frac{\partial C}{\partial w_1} = 0.8 - 0.1 \times 0.075 = 0.7925$$

- Os pesos foram ajustados numa direção que reduz o erro da rede.

Conclusão da Aula

- **Redes neurais** são modelos poderosos que aprendem a partir de dados para resolver **problemas complexos**.
- O algoritmo **backpropagation** permite treinar redes ajustando pesos com base no **erro** observado.
- Entender a **estrutura**, **função** e **treinamento** de redes neurais é fundamental para avançar em inteligência artificial.
- **Limitações** e **cuidados** na aplicação são essenciais para o uso responsável da tecnologia.

Pontos-Chave Para Lembrar

- Redes neurais aproximam **funções complexas**, processando dados por camadas de neurônios interconectados.
- **Backpropagation** usa a **regra da cadeia** para calcular **gradientes** e atualizar pesos via **gradiente descendente**.
- A escolha da **função de ativação** pode influenciar a capacidade de aprendizado da rede.
- O treinamento requer cuidado com **overfitting**, **qualidade dos dados** e **seleção de hiperparâmetros**.

Próxima Aula

- Continuaremos o estudo avançando para **Deep Learning**.
- Vamos explorar arquiteturas profundas como *Multi-Layer Perceptrons*, *Redes Convolucionais* e *Redes Recorrentes*.
- Aplicações práticas em visão computacional, processamento de linguagem natural e outras áreas.
- Técnicas para treinamento de redes profundas e desafios associados.

Referências

- Russell, S., Norvig, P., "Artificial Intelligence: A Modern Approach", 4th ed., Person, 2022.
- Duda, Richard O., "Pattern Classification", 2nd ed., Wiley, 2000.
- Muller, A. C., Guido, S., "Introduction to Machine Learning with Python A Guide for Data Scientists", O'Reilly, 2017.

Obrigado!

E-mail: fabianosoares@unb.br

LinkedIn: https://www.linkedin.com/in/fabiano-soares-06b6a821a/

Site do curso: https://www.fabianosoares.eng.br/fga0221-inteligência-artificial