

---

# Redes Neurais Artificiais

## aprendizado supervisionado (no tempo)

---



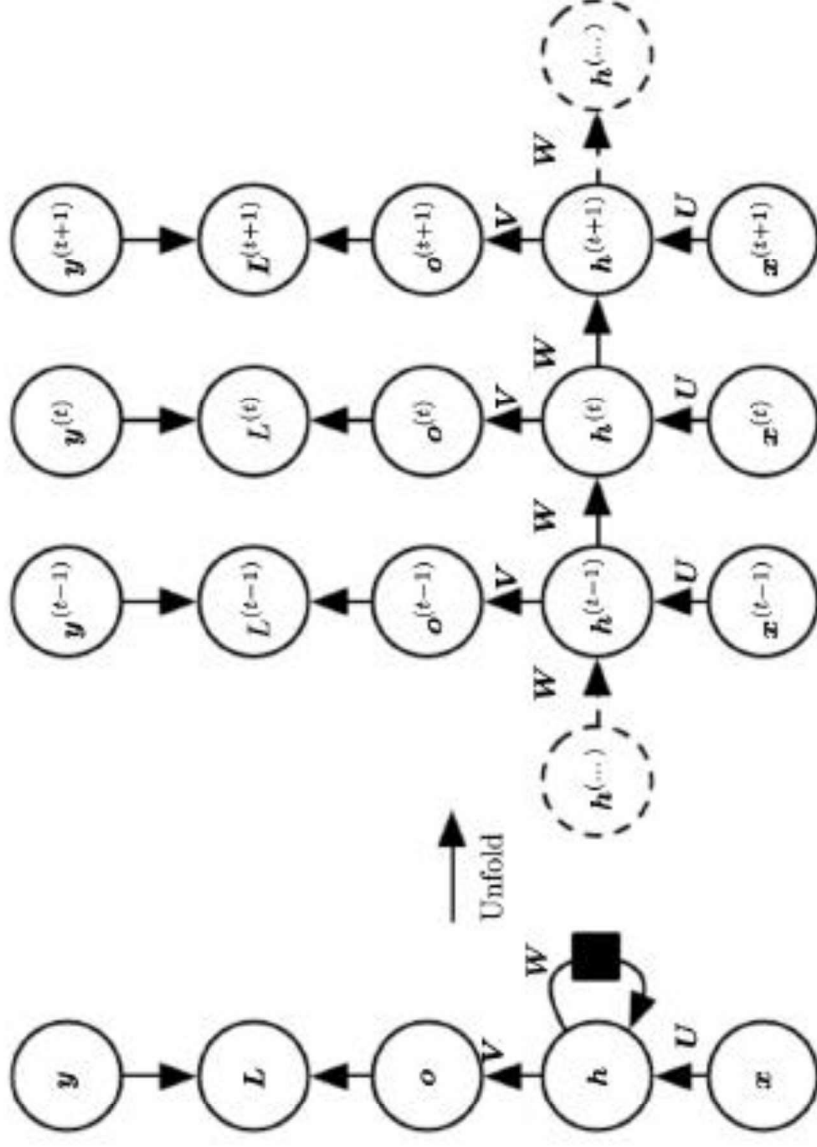
# Redes neurais recorrentes

---

- Família de redes neurais especializadas em processar dados sequenciais.
- Essas redes possuem uma arquitetura caracterizada por ciclos (recorrência) em suas conexões. Isso significa que o processamento de uma unidade (neurônio) é influenciado, de forma direta ou indireta, pela saída de um (ou mais) processamento anterior (no tempo).
- Uma área de aplicação para essa família de redes neurais bastante estudada atualmente é o processamento de língua natural, já que nessa aplicação, a ideia é processar seqüências de palavras.



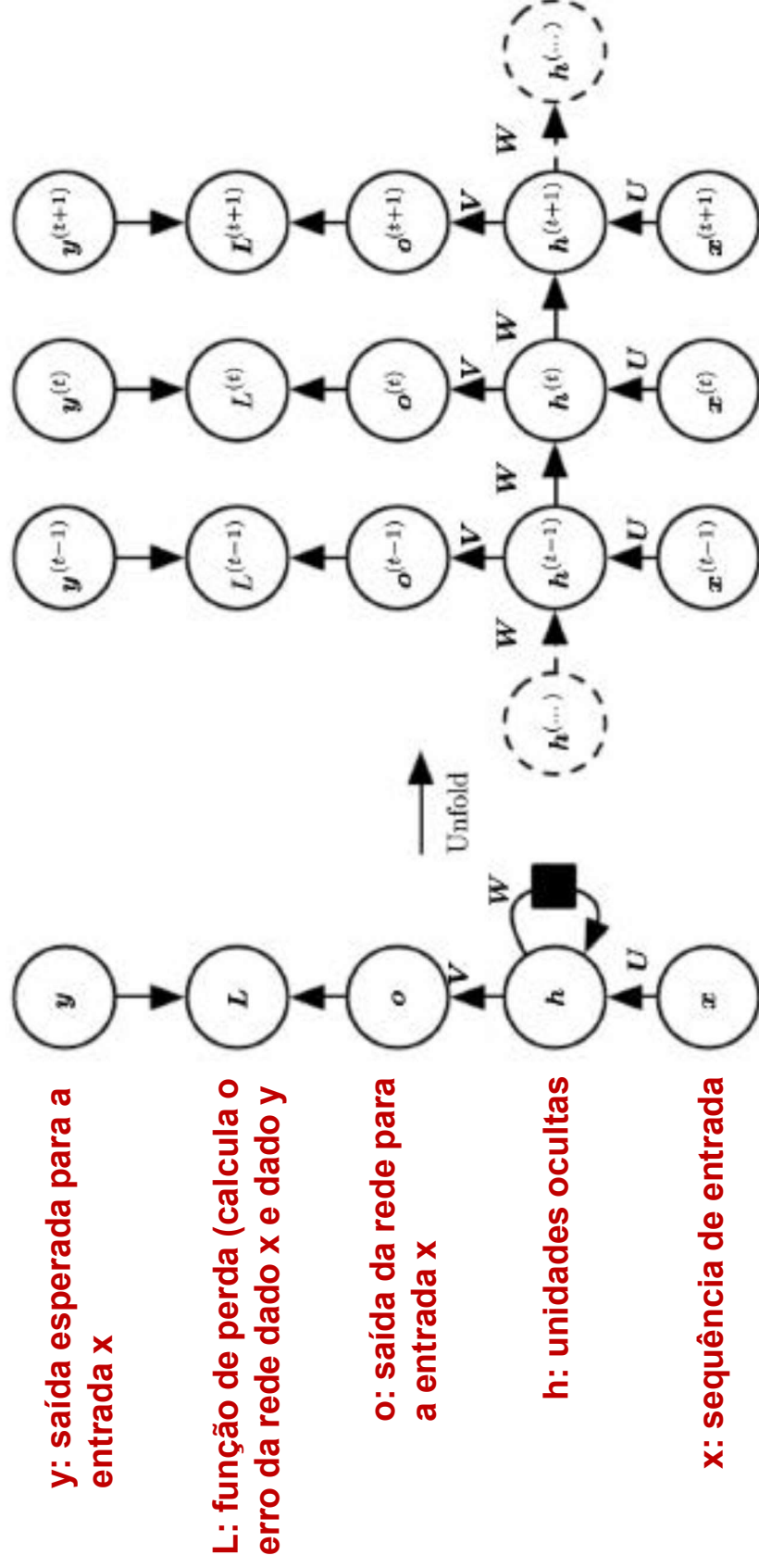
# Redes neurais recorrentes



Fonte – (GOODFELLOW; BENGIO; COURVILLE, 2016)

# Redes neurais recorrentes

(rede desenrolada no tempo)



# Redes neurais recorrentes

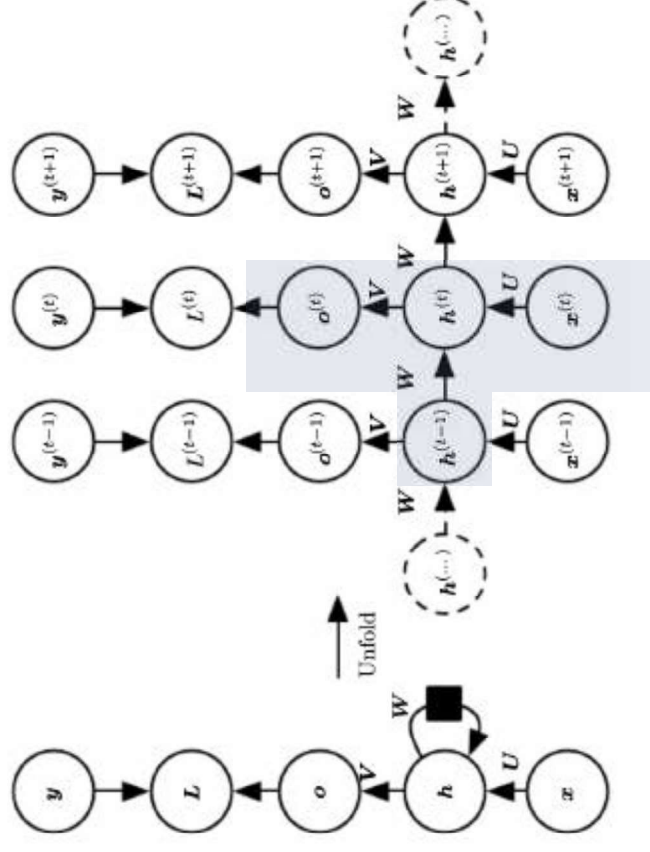
- Feedforward

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W} * \mathbf{h}^{(t-1)} + \mathbf{U} * \mathbf{x}^{(t)}$$

$$\mathbf{h}^{(t)} = \tanh(\mathbf{a}^{(t)})$$

$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V} * \mathbf{h}^{(t)}$$

$$\hat{\mathbf{y}}^{(t)} = \text{softmax}(\mathbf{o}^{(t)})$$



Fonte – (GOODFELLOW; BENGIO; COURVILLE, 2016)

# Redes neurais recorrentes

---

- **Loss**: ao final da sequência, a loss é a soma de todas as perdas, obtidas em cada tempo.
- **RBTT**: *recurrent backpropagation through time*. De forma simplificada, aplica o backpropagation tradicional a cada tempo, mas só permite a atualização dos pesos ao fim do processamento da sequência, a partir de uma totalização (média, por exemplo) de todas as alterações calculadas.
- Na fase de treinamento, o valor original da sequência é usado como entrada recorrente, e não o valor predito (**teacher forcing**).



# Rede neurais recorrentes bidirecionais

---

- Usadas para aplicações em que o contexto passado ( $t-1$ ) e o contexto futuro ( $t+1$ ) é importante para obtenção da saída.
- São a combinação de duas redes neurais recorrentes independentes: uma que “se move” do início da sequência para o final; outra que “se move” do final da sequência para o início.
- A ligação entre as duas redes pode ser feita de duas maneiras:
  - Construção do estado escondido ( $h$ ) como uma combinação do estado escondido resultante do processamento e cada uma das direções (combinação é uma opção de soma, multiplicação ou média);
  - Construção de uma saída da rede que combine as saídas das ambas as redes, também com uma soma, multiplicação ou média; ou concatenação (formando uma base para tomada de decisão).



# Memória longa de curto prazo (LSTM)

---

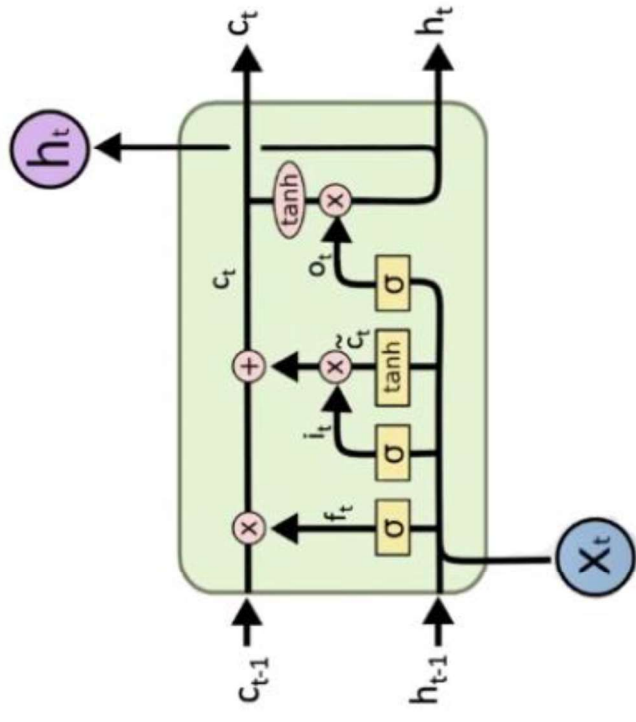
- LSTM: long short-term memory. Extensão de uma RNN.
  - Limitação de uma RNN: não consegue aprender a dependência de longo prazo de um termo em relação ao outro em uma dada sequência.
  - Uma sequência grande como entrada implica em fazer muitas (t) multiplicações pela matriz  $W$  (multiplicações sucessivas por  $W$  equivale a multiplicar por  $W^t$ ). Isso pode levar a valores muito altos ou a valores muito baixos. Valores muito baixos podem impedir o efeito do aprendizado. Valores muito altos tornam o aprendizado muito instável.
- A LSTM usa uma unidade de memória além da unidade oculta. A unidade de memória é controlada por “portas”.





# Memória longa de curto prazo (LSTM)

- A LSTM é composta por células LSTM, na qual unidades e portas são combinadas.
- As portas são vetores que, por combinarem as unidades entrada, oculta e de memória, controlam o quanto de informação é esquecida e o quanto de informação é adicionada para as tomadas de decisões posteriores.



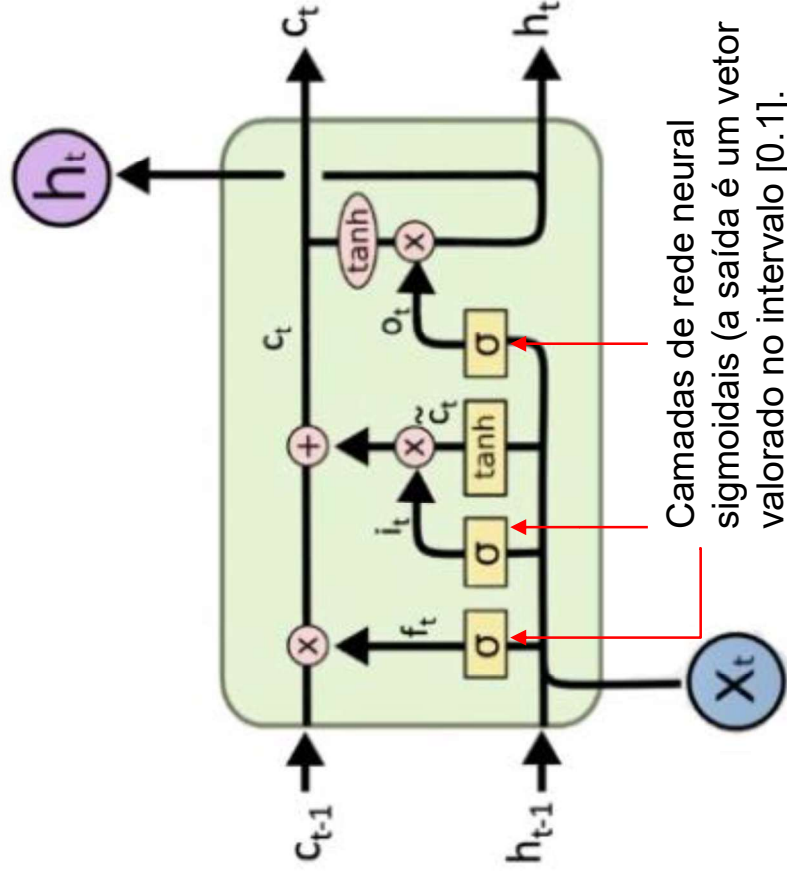
# Memória longa de curto prazo (LSTM)

- Os valores nas três portas são calculados usando:
  - a entrada atual;
  - o estado anterior.

$$f_t = \sigma(U_f * h_{t-1} + W_f * x_t)$$

$$i_t = \sigma(U_i * h_{t-1} + W_i * x_t)$$

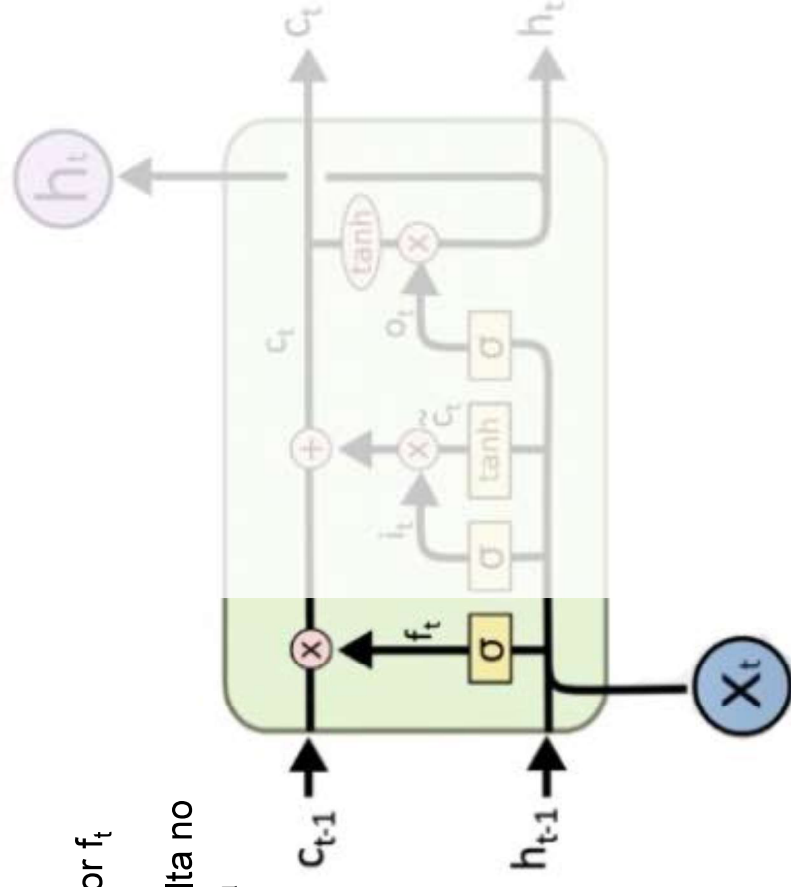
$$o_t = \sigma(U_o * h_{t-1} + W_o * x_t)$$



- A combinação destas portas com a célula de memória anterior e com um vetor de valores atualizáveis, obtém-se a próxima célula de memória e o próximo estado oculto.

# Memória longa de curto prazo (LSTM)

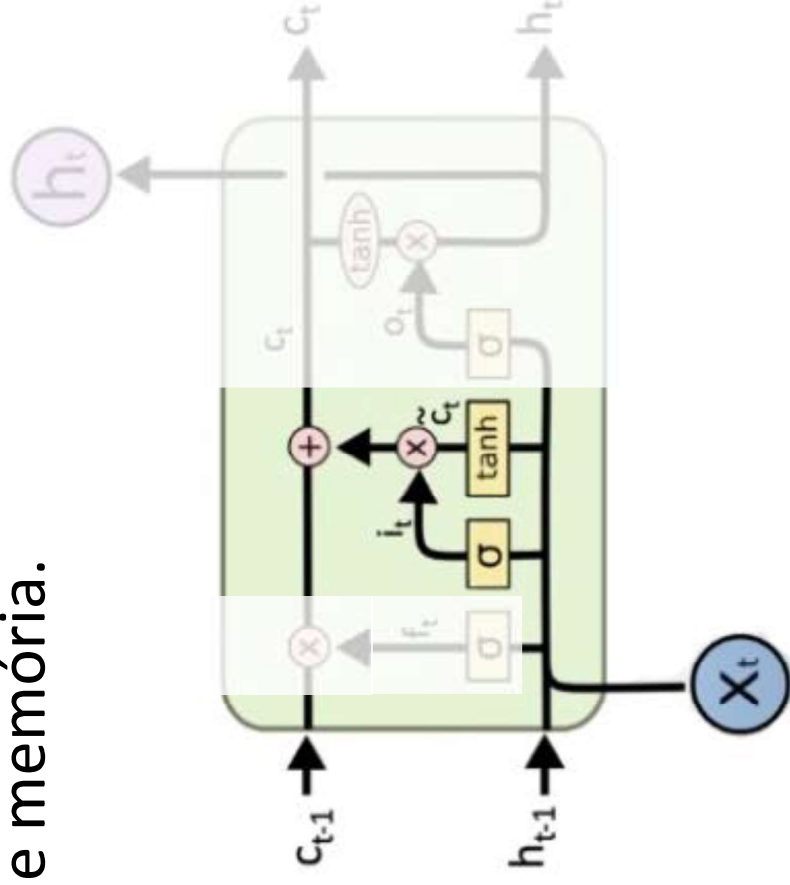
- Porta de esquecimento ( $f_t$ ): deleta informação que não é mais necessária, multiplicando a vetor produzido com a célula de memória anterior.



Valores baixos no vetor  $f_t$  indicam um tipo de esquecimento (0 resulta no esquecimento total da informação)

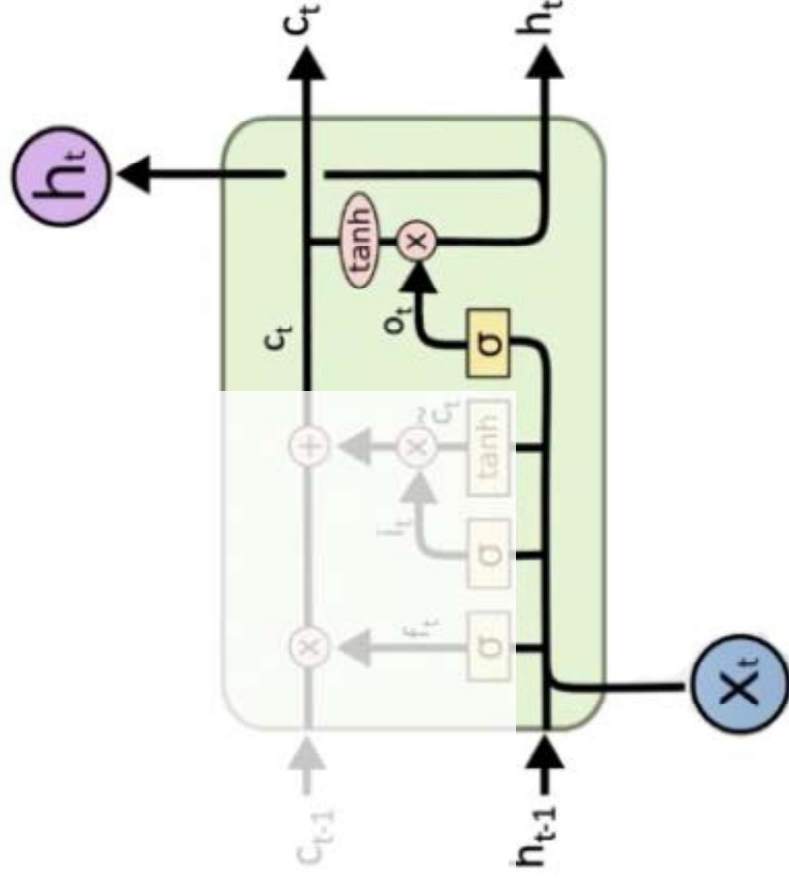
# Memória longa de curto prazo (LSTM)

- Porta de entrada ( $i_t$ ): decide qual informação é adicionada na célula de memória, multiplicando o vetor produzido como entrada ( $i_t$ ) por um vetor similar de valores candidatos ( $\tilde{c}_t$ ). O resultado é, então somado na célula de memória.



# Memória longa de curto prazo (LSTM)

- Porta de saída ( $o_t$ ): decide a informação de saída (resposta e próximo estado oculto). A saída é gerada a partir da combinação da entrada via camada sigmooidal e o que foi decidido manter na memória ( $c_t$ )



# Redes Neurais Artificiais

## com extração de representação

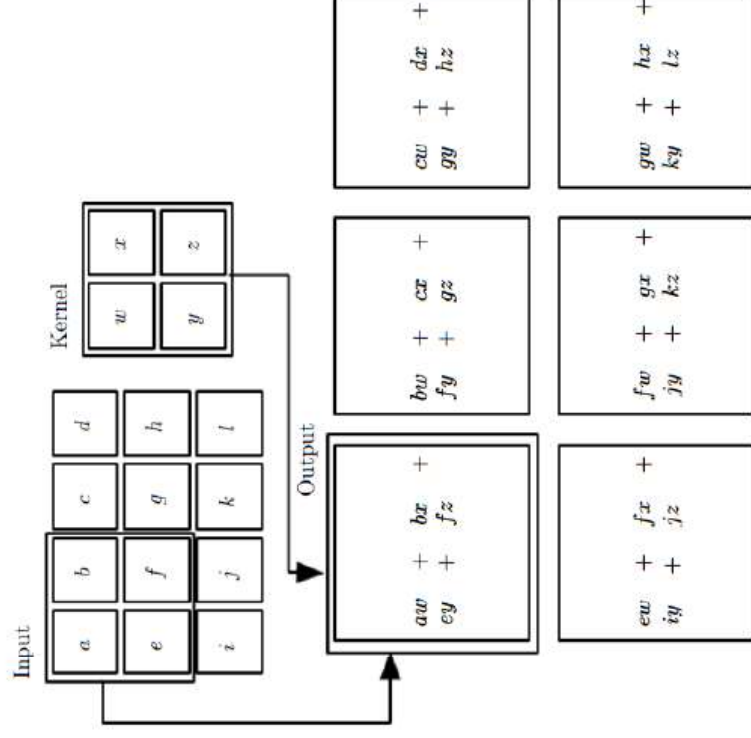


# Redes Neurais Convolucionais



# Redes Neurais Convolucionais

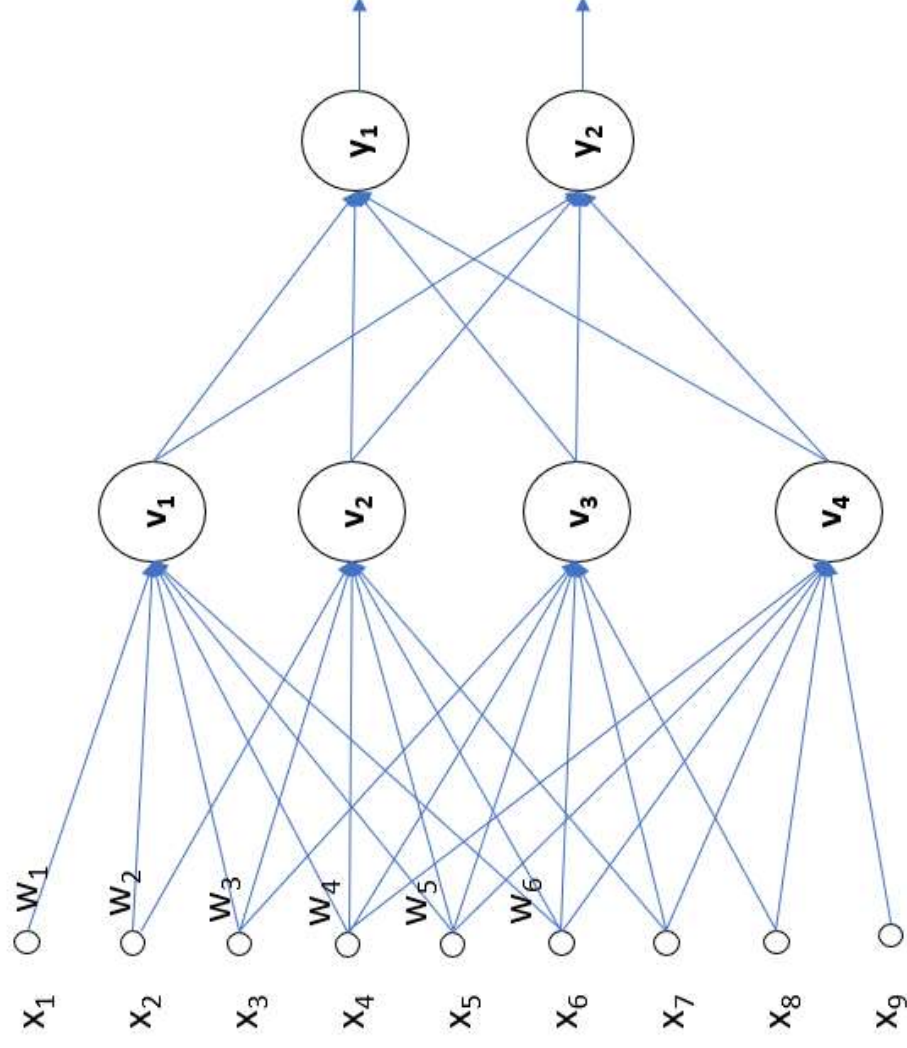
Exemplo de convolução com arrays bidimensionais



\* Goodfellow, Bengio e Courville (2016) e David Warde-Farley. Pg. 334.



# Redes Neurais Convolucionais



# Redes Neurais Convolucionais

---

Como embutir conhecimento a priori em um projeto de rede neural?

- restringindo a arquitetura da rede usando um padrão de conexões locais conhecidos como campos receptivos;
- restringindo a escolha dos pesos sinápticos usando pesos compartilhados.

A computação da entrada dos neurônios escondidos resultará em uma operação de convolução!

São redes neurais simples que usam convolução (um tipo especializado de operação linear) no lugar de multiplicação de matrizes em pelo menos uma de suas camadas.

Essa rede neural é útil para processamento de dados com *grid-like topology*: séries temporais (caso 1D) e imagens (caso 2D).



# Redes Neurais Convolucionais

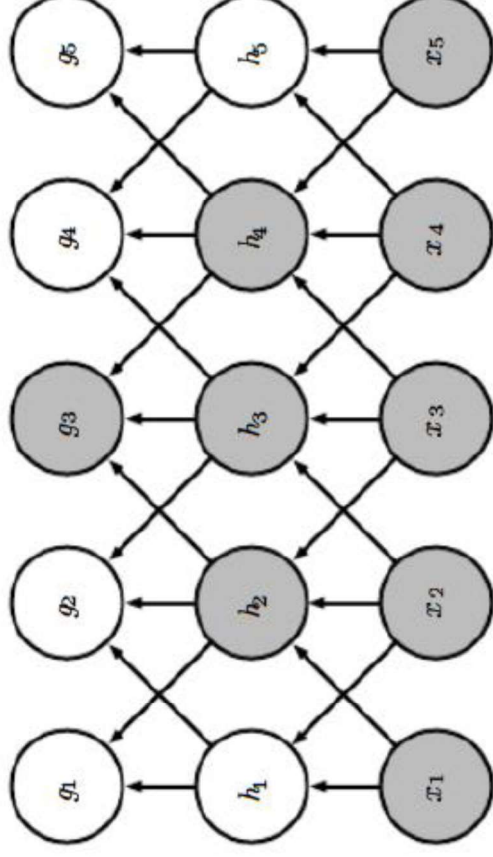
## Redes neurais convolucionais - Convolution Neural Networks - CNN

É um caso especial do Multilayer Perceptron (MLP). É um MLP projetado, por exemplo, para reconhecer formas bi-dimensionais, de maneira invariante à translação (e outras distorções a depender das operações combinadas). Ela é uma rede de aprendizado supervisionado, cuja estrutura inclui algumas restrições.

- Extração de características
- Mapeamento de características
  - invariância de translação/deslocamento (turno)
  - redução no número de parâmetros livres
- Sub-amostragem

# Redes Neurais Convolucionais

**Extração de características:** cada neurônio recebe as entradas ponderadas de seu campo receptivo local, realizando a extração de características locais. Uma vez que a característica foi extraída, a sua localização exata (em relação à entrada) se torna menos importante enquanto sua posição relativa em relação a outras características é (aproximadamente) preservada.



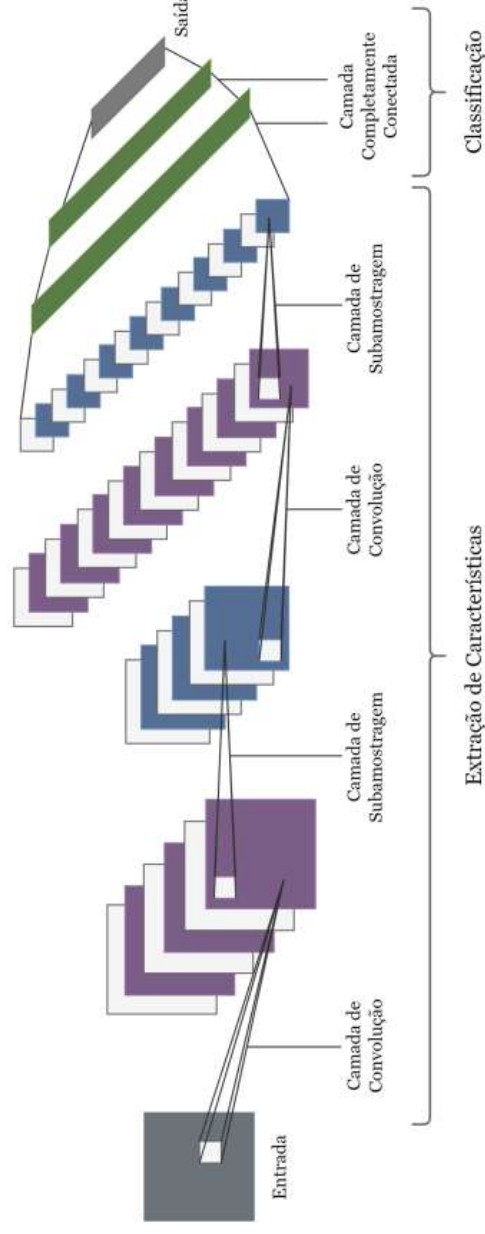
\* Goodfellow, Bengio e Courville (2016) e David Warde-Farley. Pg. 337.



# Redes Neurais Convolucionais

**Mapeamento de características:** cada camada de processamento da rede neural é composta de múltiplos mapas de características, e cada um desses mapas, na camada, compartilha os mesmos pesos sinápticos. Isso gera dois efeitos benéficos:

- o mapa de características é produzido por convolução (permitindo a invariância) seguida de uma função sigmoide (que realiza um achatamento)<sup>1</sup>
- os pesos compartilhados reduzem o número de parâmetros livres da rede



# Redes Neurais Convolucionais

---

**Sub-amostragem:** (ou pooling) Cada camada convolucional é seguida por uma camada computacional que executa uma média local e uma sub-amostragem, por meio da qual a resolução do mapa de características é reduzida. Esta operação tem o efeito de reduzir a sensibilidade da saída do mapa de características em termos de deslocamentos ou outros distorções.

Um função pooling substitui uma saída da camada anterior, em determinadas localizações, por um resumo estatístico das saídas vizinhas. Por exemplo, operação *max-pooling* retorna a saída máxima dentro de uma vizinhança retangular. Ainda é possível usar médias, normas ou distâncias no lugar da operação *max*.

---



# Redes Neurais Convolucionais - exemplo

---

Considere (ex. Haykin):

- arquitetura: uma camada de entrada, quatro camadas escondidas, uma camada de saída
  - entrada: imagens de  $28 \times 28$  neurônios sensoriais, referentes as 26 caracteres
  - o layout computacional altera entre convoluções e subamostragem (como descrito na sequência)
- 
- A primeira camada executa convolução. Ex.: seis mapas de características, cada um com  $24 \times 24$  neurônios. Cada neurônio está associado a campos receptivos de tamanho  $5 \times 5$ .





# Redes Neurais Convolucionais - exemplo

---

- A primeira camada executa convolução. Ex.: seis mapas de características, cada um com  $24 \times 24$  neurônios. Cada neurônio está associado a campos receptivos de tamanho  $5 \times 5$ .
- A segunda camada executa a sub-amostragem (ou pooling) e média local. Ex.: Ela também consiste de seis mapas de características, cada um com  $12 \times 12$  neurônios. Cada neurônio tem um campo receptivo de  $2 \times 2$ .
- A terceira camada escondida executa a segunda convolução. Ex.: Ela consiste de 16 mapas de características, cada um com  $8 \times 8$  neurônios. Cada neurônio tem conexões provenientes de vários mapas de características da camada anterior.
- A quarta executa a segunda sub-amostragem e média local. Ex.: Ela consiste de 12 mapas de características de  $4 \times 4$  neurônios.





# Redes Neurais Convolucionais - exemplo

---

- A quarta executa a segunda sub-amostragem e média local. Ex.: Ela consiste de 12 mapas de características de  $4 \times 4$  neurônios.
- A camada de saída é o estágio final de convolução. Ex.: Um MLP de duas camadas em que cada neurônio na saída é associado a um dos 26 caracteres possíveis. Cada neurônio da “entrada” é associado ao campo receptivo de tamanho  $4 \times 4$ , com pesos sinápticos provenientes de vários mapas de características da camada anterior.



# Autoencoders

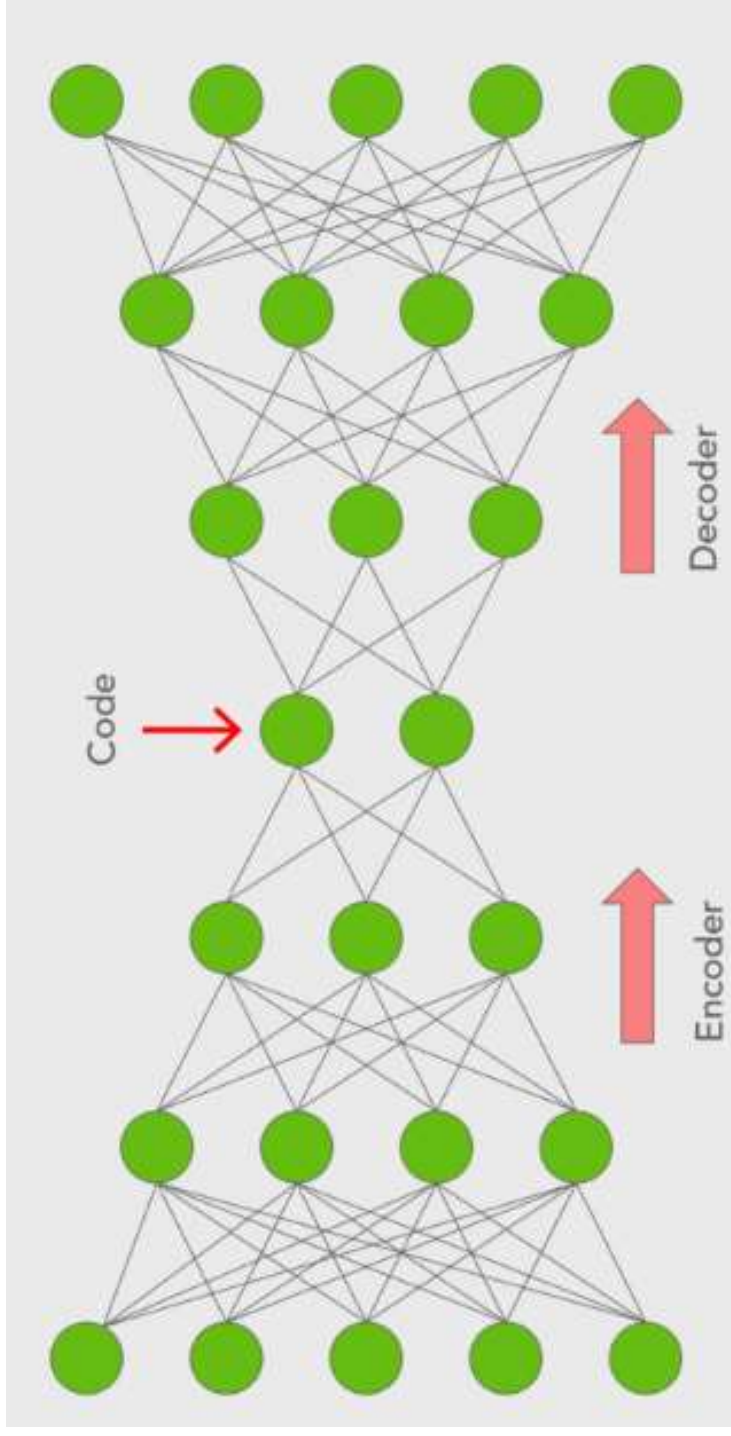


# Autoencoders

- Rede neural treinada para copiar as entradas nas saídas.
- Trata-se de uma tarefa comumente conhecida como “reconstrução”.
- Internamente, existe uma camada escondida que descreve a “codificação” usada para representar a entrada.
- **Encoder:** uma função do tipo mapeamento  $h = f(x)$ ;
- **Decoder:** uma função de reconstrução  $r = g(h)$ .

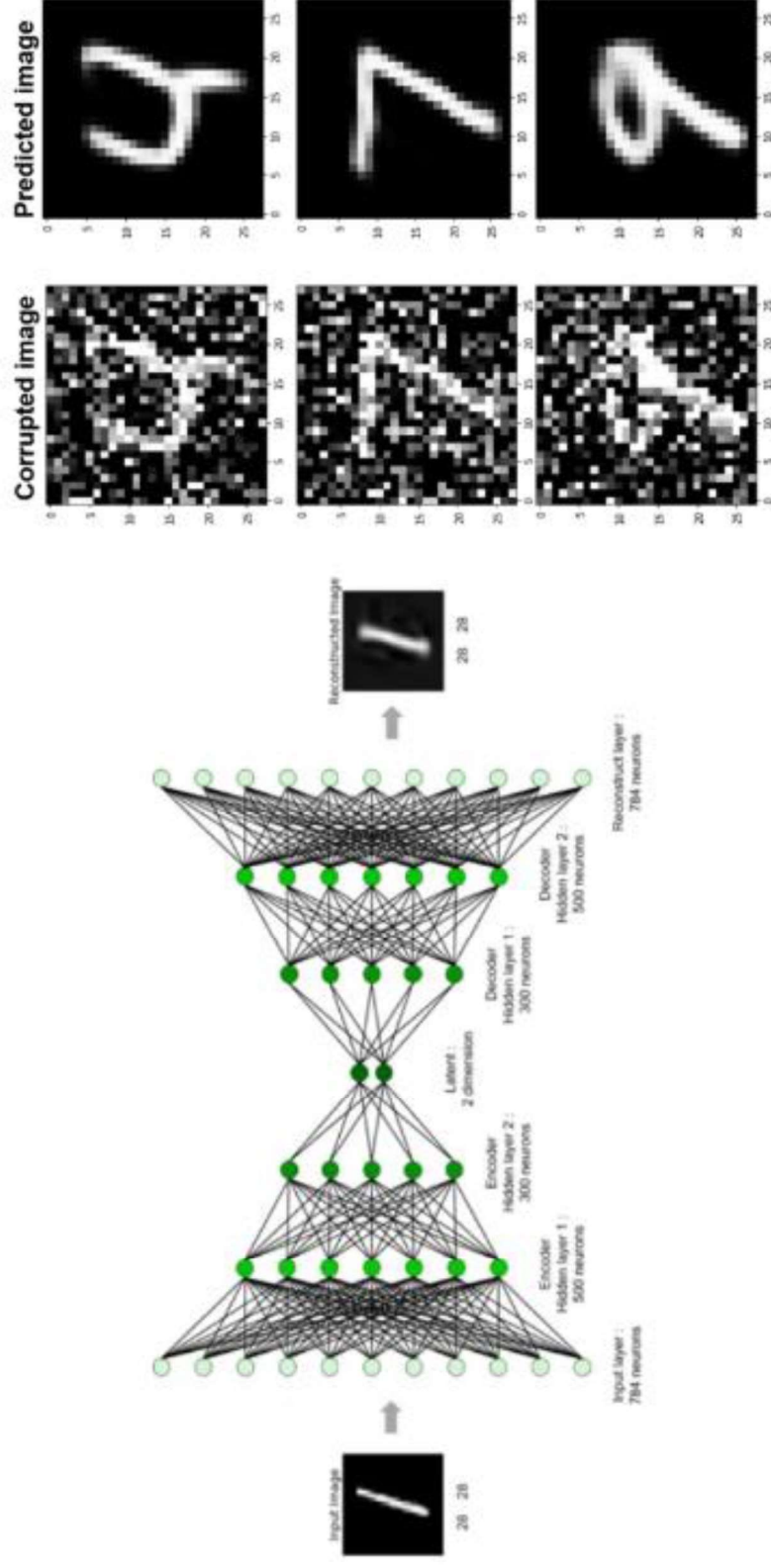
Se um autoencoder implementar a função identidade --  $g(f(x)) = x$  -- então ele não será útil. O que queremos são autoencoders projetados para não serem capazes de aprender a copiar perfeitamente.

# Autoencoders



**Treinamento: por retropropagação do erro e minimização do gradiente – mas é conhecido como não supervisionado.**

# Autoencoders – aplicação



# Autoencoders

---

- **Autencoder subcompleto ou subdimensionado (undercomplete)**: autoencoder em que a dimensão da codificação (camada  $h$ ) é menor que a dimensão da entrada.
- Neste caso, podemos estar interessados na ou compactação da entrada ou na extração de características (apenas) e a função identidade será bem-vinda no treinamento da rede.
- **Cuidado**: se o autoencoder tem muita capacidade (encoder e decoder não lineares) ele pode ter sucesso no treinamento (aprender a função identidade) mas não ser capaz de fazer uma boa extração de características.



# Autoencoders

---

- **Autoencoders regularizados**: autoencoder regularizados fornecem um caminho para prevenir que o treinamento do autoencoder deixe de aprender características interessantes (seja a camada  $h$  menor, de igual tamanho, ou maior que a camada de entrada). Também ajuda na robustez em relação a entradas ruidosas e faltantes.

Regularização: conjunto de técnicas que ajudam a evitar o sobreajuste da rede. Qualquer ajuste que façamos no treinamento com o objetivo de reduzir o erro de generalização sem prejudicar o erro de treinamento.

- **Autoencoder de remoção de ruído (denoising)**: recebe um dado corrompido como entrada e é treinado para predizer o dado original, não corrompido.



# Redes Neurais Artificiais

## aprendizado profundo





# Aprendizado profundo

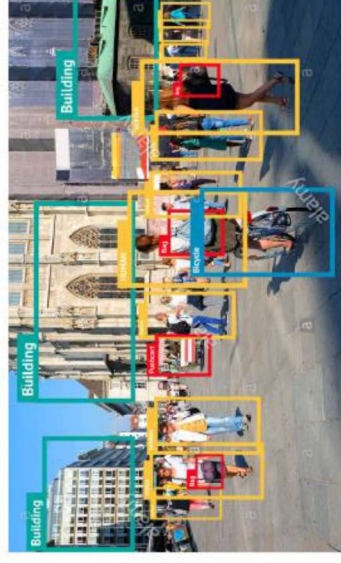
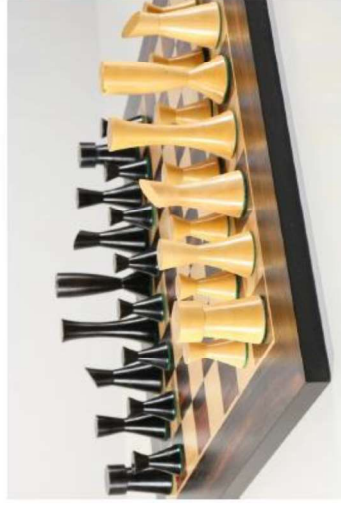
## Desafio na IA

Fornecer soluções para tarefas que são facilmente resolvidas por humanos mas que são difíceis de descrever formalmente – problemas cuja solução é intuitiva e que parece automática: reconhecimento de palavras faladas, faces, imagens ....

- **Aprendizado:** aprender a partir de experiências, obter conhecimento a partir da experiência, evitando que humanos precisem especificar todo o conhecimento que o computador precisa ter para resolver um problema.
- **Hierarquia de conceitos:** entender o mundo em termos de uma hierarquia de conceitos na qual cada conceito é definido em termos de suas relações com conceitos mais simples. Se nós desenharmos um grafo mostrando conceitos construídos sobre outros conceitos teremos um gráfico profundo, com muitas camadas.

# Aprendizado profundo

- O caso do **jogo de xadrez** representa um problema que exige pouco conhecimento sobre o mundo, e que possui um conjunto de regras objetivo e finito.
- O caso de **reconhecimento de imagens**, ou de identificação de objetos em uma imagem requer muito conhecimento sobre o mundo, e é um conhecimento subjetivo e intuitivo.



A questão chave é:

Como incorporar conhecimento e informação em um programa de computador!

# Aprendizado profundo

- Algoritmos de aprendizado de máquina dependem fortemente da representação dos dados - a forma como os dados são apresentados ao algoritmo.
- Cada pedaço de informação incluída na representação dos dados é conhecida como *feature* (característica descritiva).
- O algoritmo de aprendizado de máquina (tradicional) não tem o controle sobre quais características são fornecidas a ele, embora, a depender do algoritmo usado, determinadas características influenciarão mais ou menos a resposta, ou modelo, gerada.

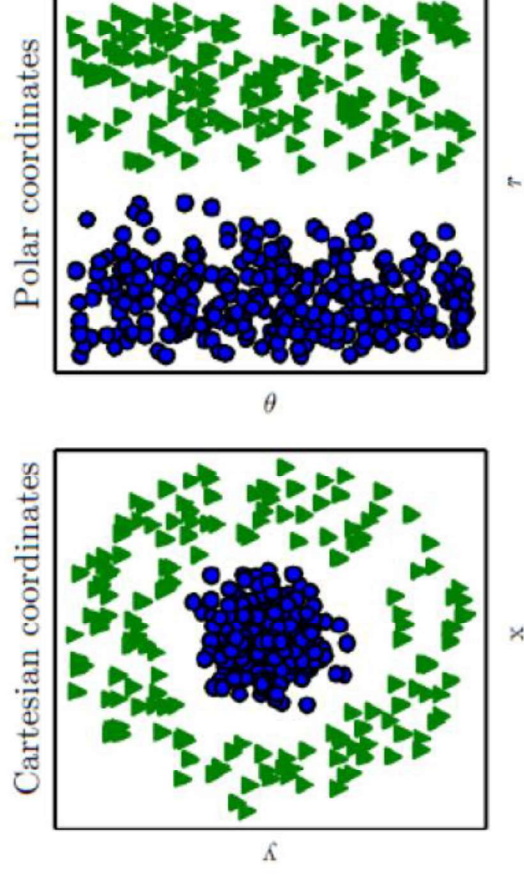
**Pense:** fazer contas com algarismos arábicos e fazer contas com algarismos romanos.

$$\begin{array}{r} 2009 \overline{) 19} \\ 109 \phantom{00} \\ \underline{105} \phantom{00} \\ 95 \phantom{00} \\ \underline{91} \phantom{00} \\ 14 \end{array}$$

$$\begin{array}{r} \text{MCCXXIII} + \text{MCXIV} \\ = \text{MCXXIII} + \text{MCXIII} \\ + \begin{array}{r} \text{M} \quad \text{CC} \quad \text{XX} \quad \text{III} \\ \text{M} \quad \text{C} \quad \text{X} \quad \text{III} \\ \hline \text{MM} \quad \text{CCC} \quad \text{XXX} \quad \text{IIIII} \end{array} \\ = \text{MMCCCXXXVII} \end{array}$$

# Aprendizado profundo

Representação com coordenadas cartesianas e representação com coordenadas polares.



\* Goodfellow, Bengio e Courville (2016) e David Warde-Farley. Pg. 4.

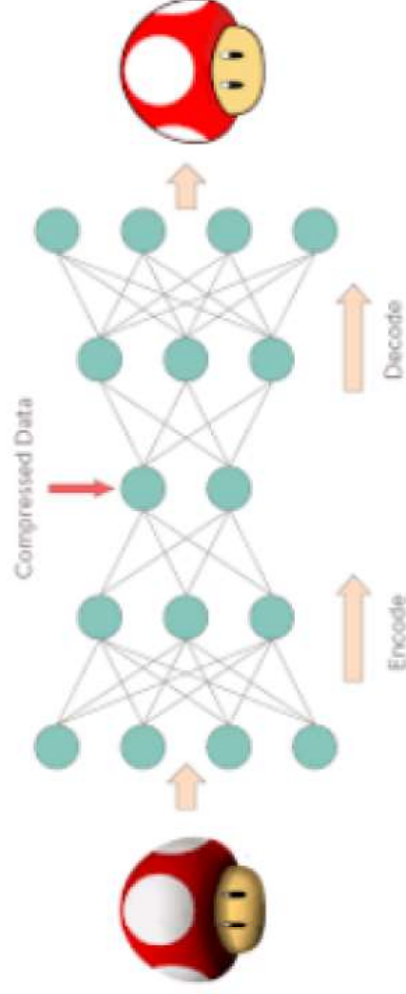
Como você descreveria um carro? Usando as rodas? Como descrever as rodas? Que forma elas assumem em uma imagem?



# Aprendizado profundo

Uma solução para esse problema é usar o aprendizado de máquina tanto para **descobrir o mapeamento da representação (entrada) para a saída**, quanto a **representação em si: *representation learning***.

O exemplo clássico é o *autoencoder*: uma combinação de uma função de codificação - que converte os dados em uma representação, com uma função de decodificação - que converte a representação de volta ao formato original.




# Aprendizado profundo

---

Aprendizado profundo atua fortemente na questão da representação, construindo conceitos complexos a partir de conceitos simples. O conceito referente à imagem de uma pessoa pode ser representado usando conceitos simples como "cantos e cortornos", que por sua vez podem ser definidos em termos de arestas.

## Aprendizado profundo e o Multilayer Perceptron (MLP)

O MLP constrói uma função matemática que mapeia um conjunto de valores de entrada em um conjunto de valores de saída. A função é formada pela composição de muitas funções mais simples: nós podemos encarar o resultado de cada função simples como uma nova representação para a sua entrada.



# Aprendizado profundo

## O que é profundo?

- número de instruções sequenciais que devem ser executadas ao passar pela arquitetura: o comprimento do caminho mais longo no fluxo de trabalho para produzir uma saída dado uma entrada. Isso depende da granularidade do passo computacional assumido pela linguagem de representação.
- profundidade do grafo que descreve como os conceitos estão relacionados com outros conceitos. Porém, quanta computação é necessária para processar um conceito pode variar.

então .... ?

Não há um consenso sobre qual profundidade é necessária para que um modelo seja qualificado como profundo.

# Aprendizado profundo

---

Os autores do livro resumem que:

Aprendizado profundo é ....

... um tipo de aprendizado de máquina que possui muito poder e muita flexibilidade porque aprende a representar o mundo como uma hierarquia de conceitos aninhados, sendo que cada conceito é definido em termos de conceitos mais simples, e como representações mais abstratas computadas em termos de outras menos abstratas.

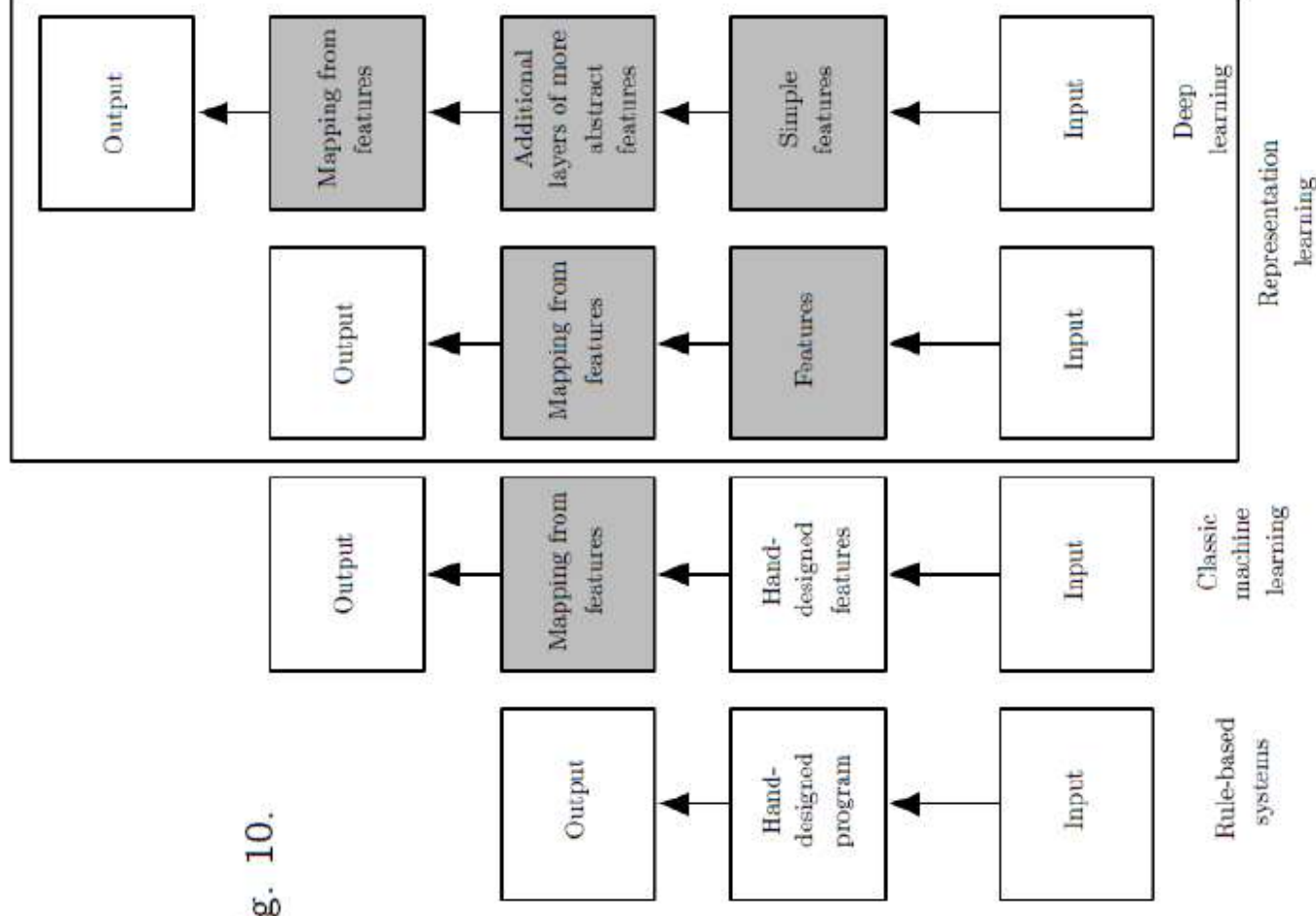
\* Goodfellow, Bengio e Courville (2016)





# Aprendizado profundo

\* Goodfellow, Bengio e Courville (2016) Pg. 10.



---

# Redes Neurais Artificiais

aprendizado profundo – hot topics

---



# Modelo codificador-decodificador

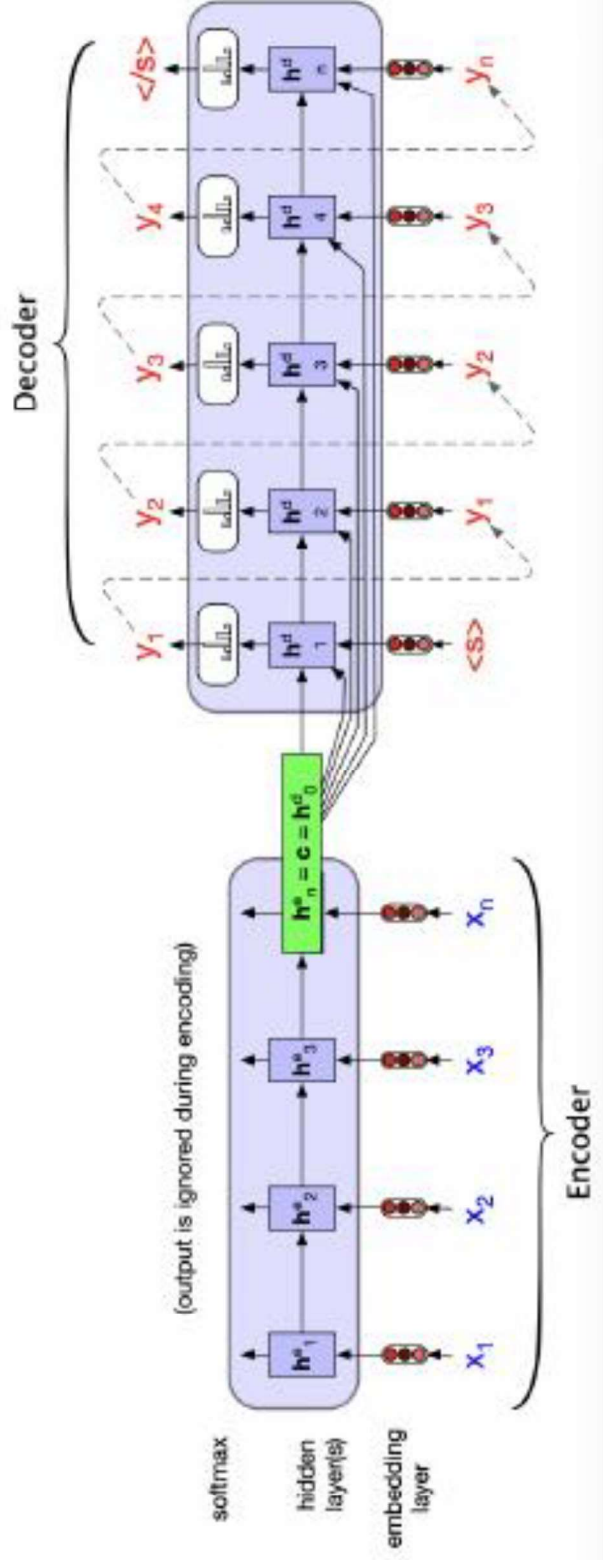
---

- Também conhecido como modelo sequência-para-sequência (*sequence-to-sequence*): mapeia uma sequência de entrada para uma sequência de saída.
  - Componentes:
    - **Codificador**: processa uma sequência de entrada e gera uma representação contextualizada desta sequência.
    - **Contexto**: o último estado do codificador.
    - **Decodificador**: utiliza o contexto como estado inicial e gera a sequência de saída, de forma autorregressiva, até a marcação final da sequência.
- 



# Modelo codificador-decodificador

- **Codificador:** processa uma sequência de entrada ( $x$ ) e gera uma representação contextualizada ( $h$ ) desta sequência.
- **Contexto:** o último estado do codificador ( $h = c$ ).
- **Decodificador:** utiliza o contexto como estado inicial e gera a sequência de saída, de forma autorregressiva, até a marcação final da sequência.

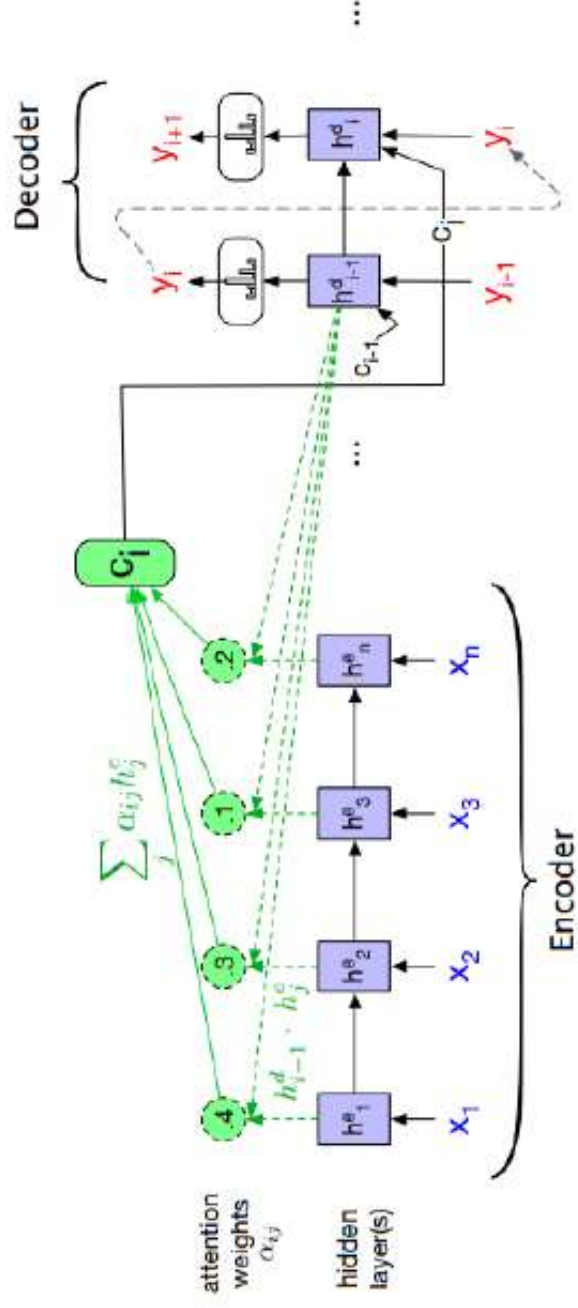


Fonte – (JURAFSKY; MARTIN, 2023)

Modelo implementado com redes recorrentes (poderia ser usada as LSTMS ou os Transformers)

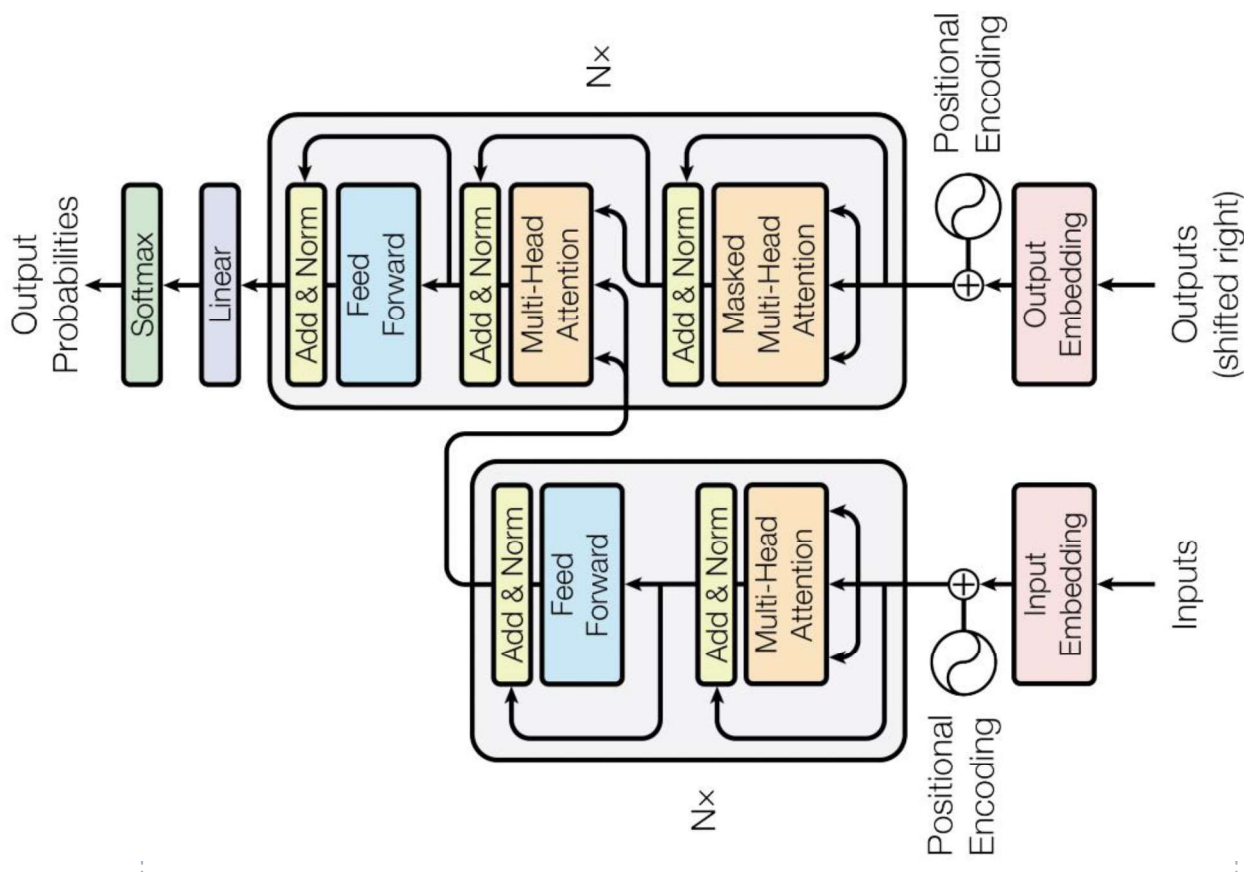
# Mecanismo de atenção

- No modelo codificador-decodificador o contexto é o último estado oculto do codificador: ele engloba “toda” a informação sobre a sequência, se tornando um gargalo e perdendo detalhes da informação original.
- **Solução:** mecanismos de atenção: todos os estados ocultos serão utilizados para compor o contexto  $c$ .



# Transformers

- Arquitetura que NÃO utiliza recorrência na obtenção de representações. Tem no mecanismo de atenção o seu principal bloco. Também utiliza o modelo codificador-decodificador. Os blocos ou camadas da arquitetura são chamados de Transformers.



# Transformers

---

- **Codificador:** cada bloco é dividido em duas subcamadas – multihead attention e rede neural feedforward. Entre as camadas existe:
  - uma camada residual: que soma a entrada à saída da camada anterior para ajudar a prevenir o problema da dissipação do gradiente.
  - uma camada de normalização: para estabilizar a saída da rede após a aplicação do resíduo.
- **Decodificador:** a subcamada multihead attention trabalha sobre a saída do codificador. Além disso, há uma camada de multihead attention com máscara para não permitir a verificação de posições futuras em uma sequência.



# Transformers

---

- **Multi-head attention:** implementa a auto-atenção, inovando no fato que cada posição  $i$  de uma sequência de entrada receberá uma “atenção” diferente (matriz de pesos diferentes).
- **Representação de posição em uma sequência:** diferentemente da rede neural recorrente, na qual a representação de ordem para uma sequência está implícita na estrutura do modelo, na arquitetura Transformers a informação não existe. A informação precisa ser fornecida ou aprendida.

