MÉTODOS E MODELOS AVANÇADOS EM CIÊNCIA DE DADOS

Aula 04 - Redes Neurais Recorrentes

(*Long-short Term Memories - LSTMs*)

Prof. Rafael G. **Mantovani**

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Universidade Tecnológica Federal do Paraná (UTFPR)
Especialização em Ciência de Dados

Roteiro

1. Introdução
2. Redes Neurais Recorrentes (*RNNs*)
3. Tipos de RNNs
4. Treinamento de RNNs
5. *LSTMs*
6. Síntese / Próximas Aulas
7. Referências

Roteiro

1. **Introdução**
2. **Redes Neurais Recorrentes (*RNNs*)**
3. **Tipos de RNNs**
4. **Treinamento de RNNs**
5. ***LSTMs***
6. **Síntese / Próximas Aulas**
7. **Referências**

Introdução

- Prever o futuro é algo que fazemos o tempo todo …

Introdução

- Prever o futuro é algo que fazemos o tempo todo …

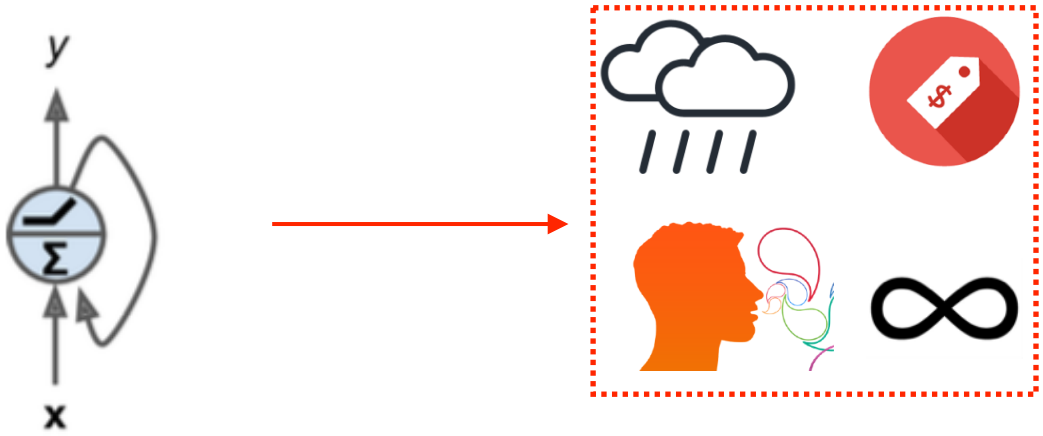
se vai chover ou não

preços de produtos/ações

fala/pensamento de alguém

Introdução

Redes Neurais Recorrentes
*(Recurrent Neural Networks - **RNNs**)*



RNNs compões uma classe de RNAs que podem prever o futuro (até certo ponto ...)
- Séries temporais **(time series)**

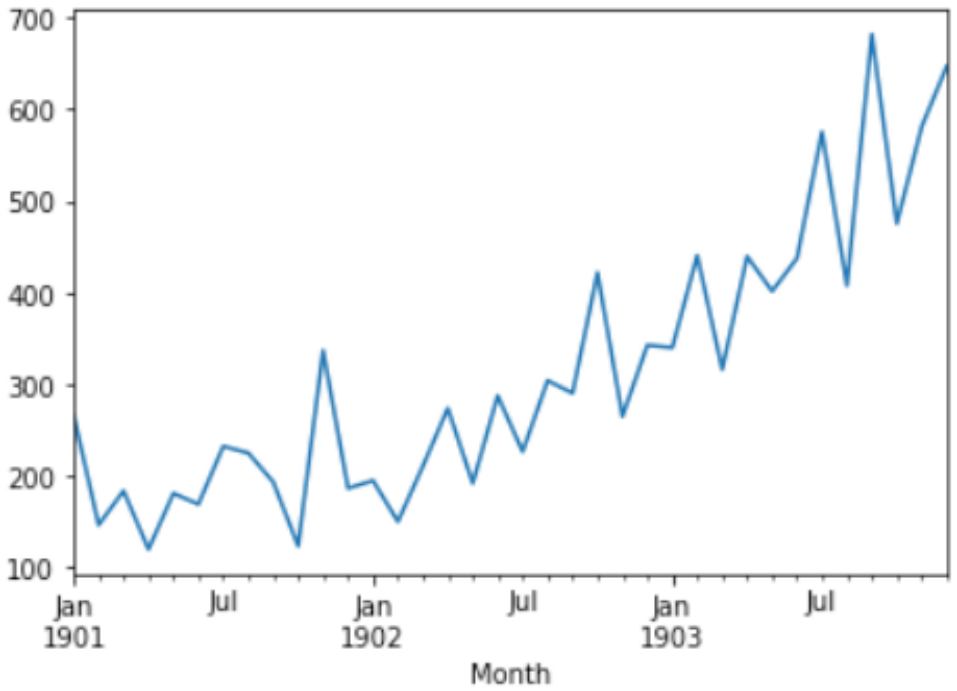
Introdução

- **Série temporal:**
  - é um conjunto sequencial de pontos de dados, constituída por uma ou várias variáveis ao longo do tempo, organizados em uma ordem cronológica adequada.

- Tipos:
  - **univariada**: única variável de observação
  - **multivariada**: mais de uma variável

Introdução

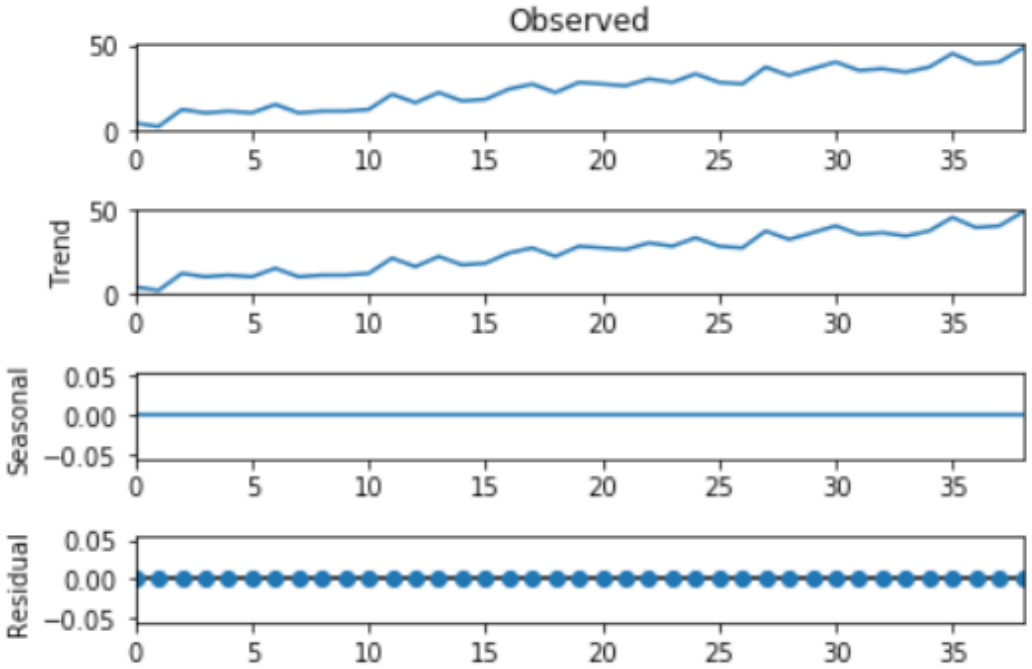
```
Name: Sales, dtype: float64
```



Fluxo temporal

Introdução

- **Características:**



Valores originais

Tendência dos valores

Sazonalidade

Resíduos = original - trend - seasonal

Introdução

- **RNNs** podem analisar:
  - séries temporais
  - frases/sentenças (linguagem natural)
  - imagens
  - documentos/textos
  - arquivos de áudio

Introdução

- **Duas maiores dificuldades/problemas:**
  - Gradientes Instáveis
    - dropout
    - normalização
    - outros otimizadores
  - memória limitada (*short-term memory*)
    - usando células/unidades mais robustas
    - LSTMs

Roteiro

1. Introdução
2. Redes Neurais Recorrentes (*RNNs*)
3. Tipos de RNNs
4. Treinamento de RNNs
5. *LSTMs*
6. Síntese / Próximas Aulas
7. Referências

RNNs

neurônios recorrentes

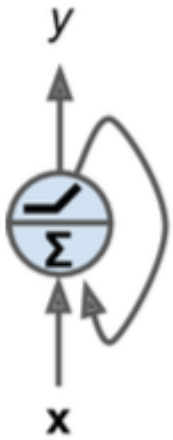
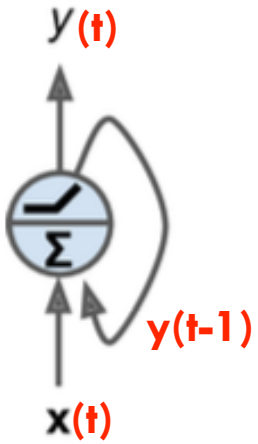


Figura de: Aurélien Gerón (2019)

RNNs

neurônios recorrentes



- Similar a uma rede *feedforward*, mas também tem **conexões para trás**
- a cada instante de tempo (t), *frame*, a rede recebe inputs x(t), como também as entradas da iteração anterior y(t-1)
- Como não existe saída anterior para t = 0, y(0) = 0

Figura de: Aurélien Gerón (2019)

RNNs

- RNNs → estendidas ao longo do **tempo**
  - mesmo neurônio representado uma ver por unidade de tempo

RNNs

- RNNs → estendidas ao longo do **tempo**
  - mesmo neurônio representado uma ver por unidade de tempo

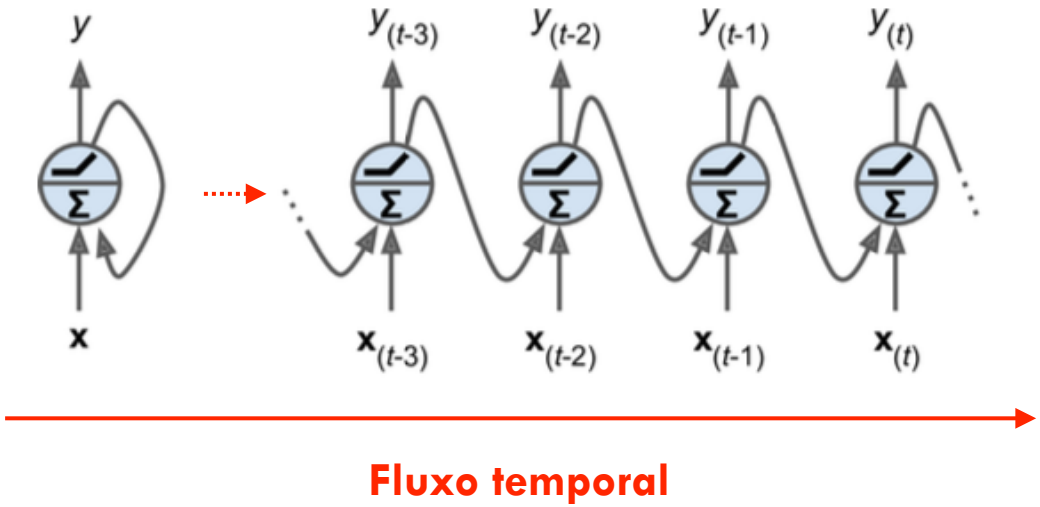


Figura de: Aurélien Gerón (2019)

RNNs

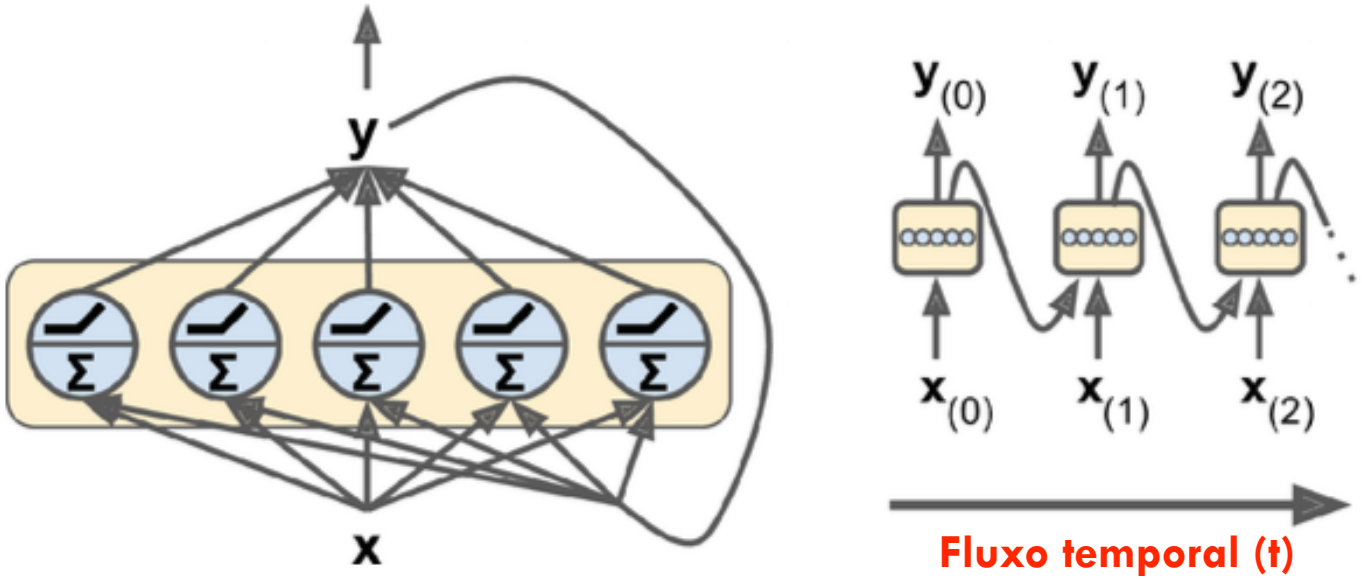


Figura de: Aurélien Gerón (2019)

RNNs

- Podemos também criar **camadas** com neurônios recorrentes

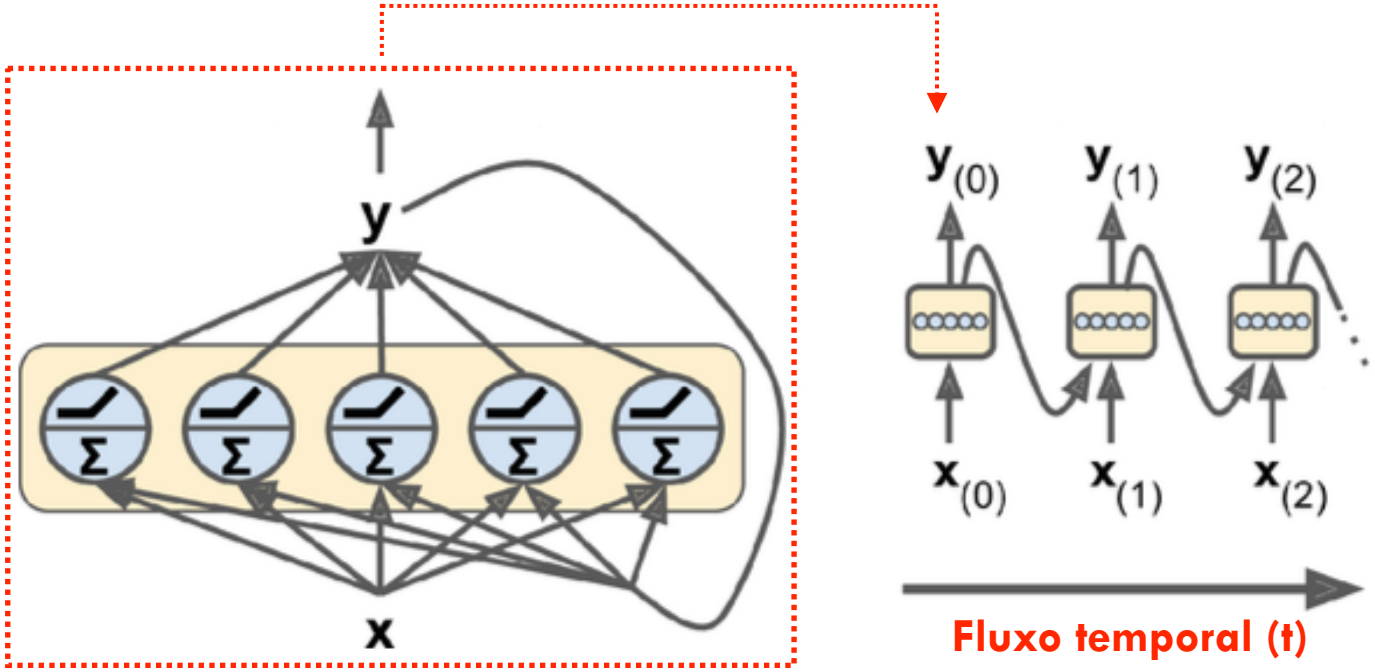
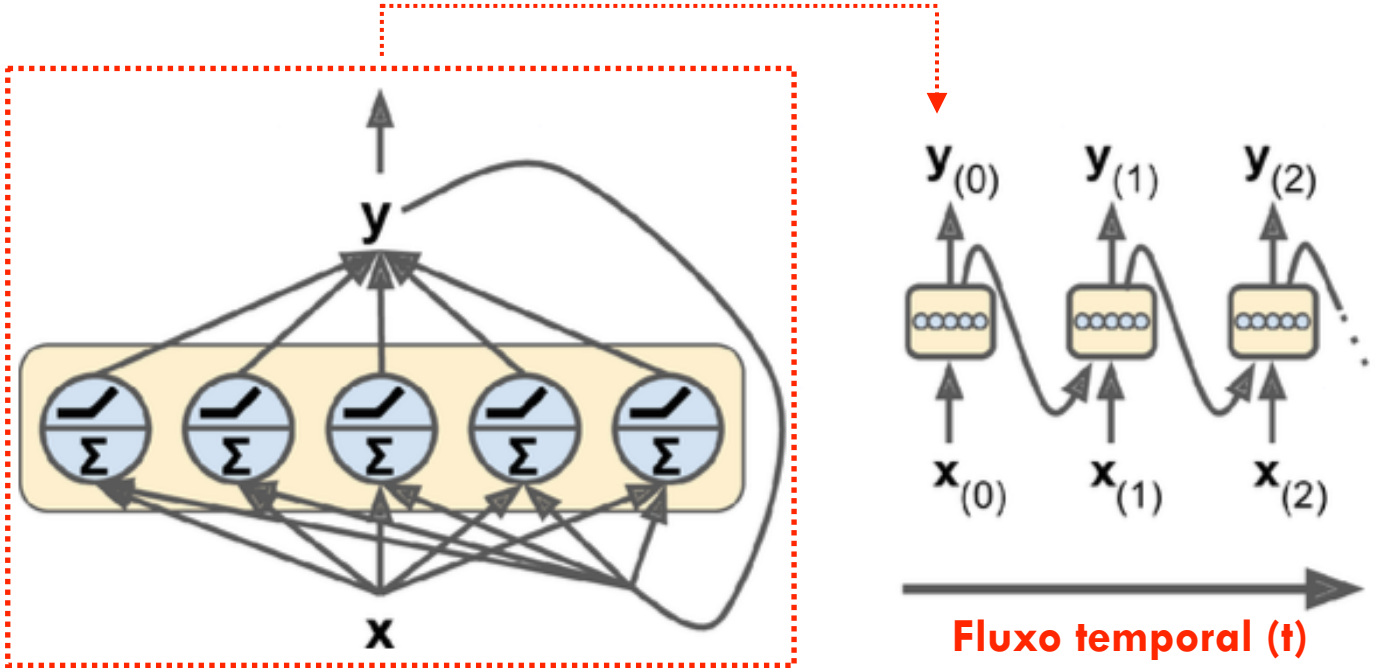


Figura de: Aurélien Gerón (2019)

RNNs

- Podemos também criar **camadas** com neurônios recorrentes



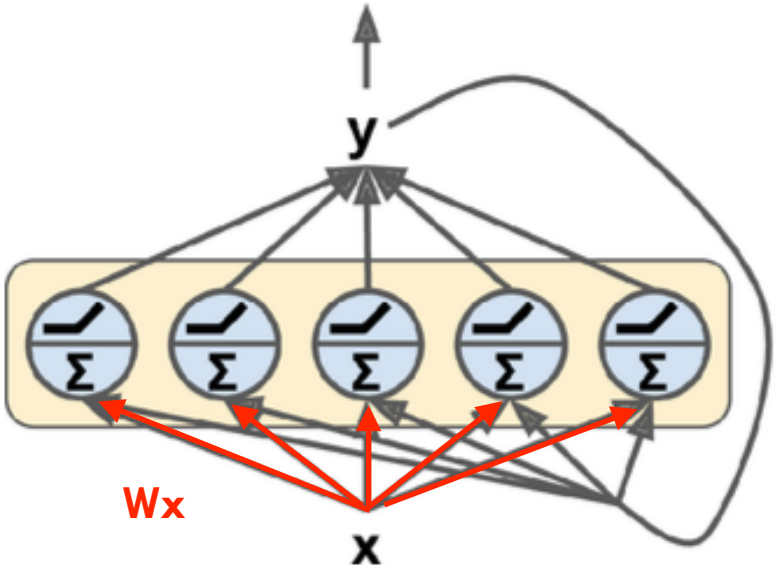
- **inputs:** vetores x(t), y(t-1) →para todo neurônio

Figura de: Aurélien Gerón (2019)

RNNs

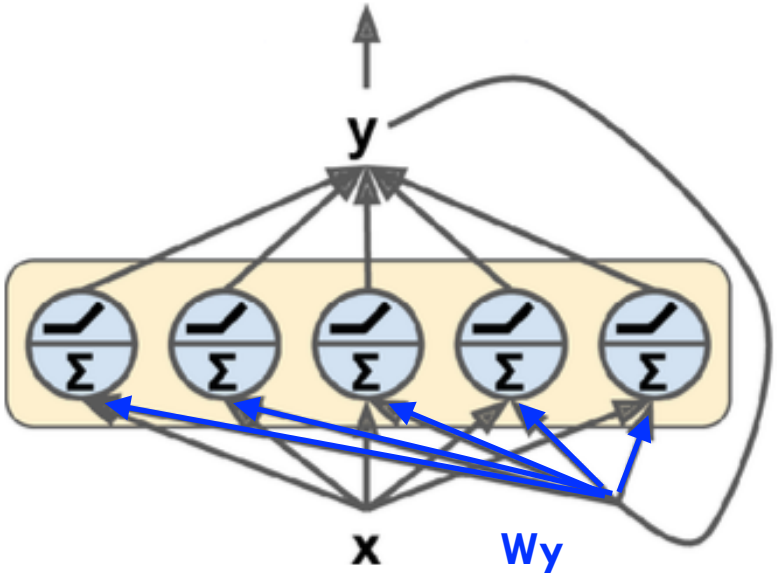
- Cada neurônio recorrente tem **dois conjuntos de pesos:**
  - ***wx***: um conjunto de pesos ligados aos **inputs** x(t)
  - ***wy***: outro conjunto ligado às **saídas** do instante de tempo anterior, y(t-1)
  - representação matricial: **Wx, Wy**

RNNs



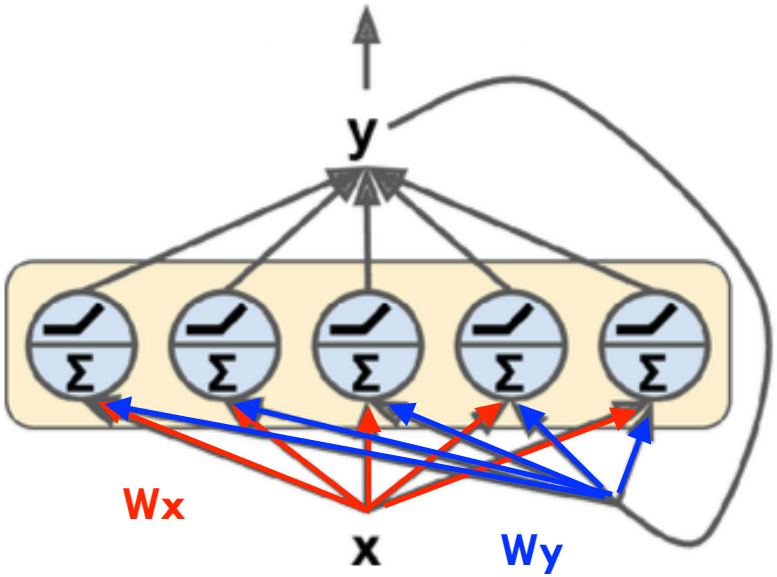
- **Wx**: conexão das inputs **x(t)** com os neurônios recorrentes

RNNs



- **Wy**: conexão das saídas anteriores, **y(t-1)** com os neurônios recorrentes

RNNs



- **Wx**: conexão das inputs **x(t)** com os neurônios recorrentes

- **Wy**: conexão das saídas anteriores, **y(t-1)** com os neurônios recorrentes

RNNs

- Ativação para única instância:

$$y_{(t)} = \phi(W_x^T x_{(t)} + W_y^T y_{(t-1)} + b)$$

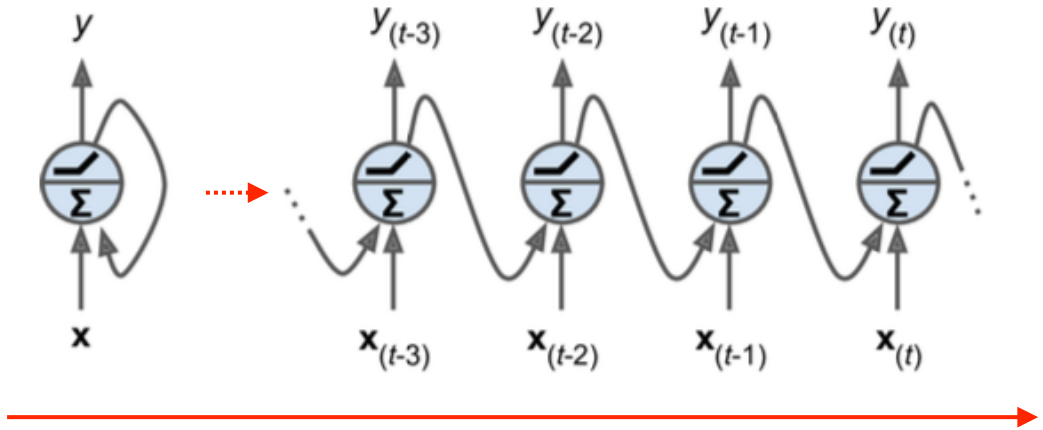
- Ativação para mini-batch:

$$Y_{(t)} = \phi([X_{(t)}\ Y_{(t-1)}]\ W + b)$$

RNNs

- com:
  - Y(t): matriz contendo as saídas da rede
  - X(t): matriz contendo todas as entradas para todas as instâncias
  - Wx: matriz contendo os pesos para as entradas (instante de tempo corrente)
  - ϕ: função de ativação
  - b: vetor de bias, com o bias para cada neurônio

RNNs



- Y(t) é uma função de X(t) e Y(t-1), que é função de X(t-1) e Y(t-2), e assim por diante.
- Logo, Y(t) é função de **todos** valores de X desde t = 0

Figura de: Aurélien Gerón (2019)

RNNs

- **Células de Memória:**
  - Saída do neurônio em um tempo **t** é uma função das entradas dos passos anteriores
  - forma de "**memória**"
    - parte da rede que preserva algum estado ao longo do tempo
    - estado da célula: $h(t) = f(x(t), h(t-1))$
  - basic cells, y(t) = h(t), but for more complex cells this is not always the case

RNNs

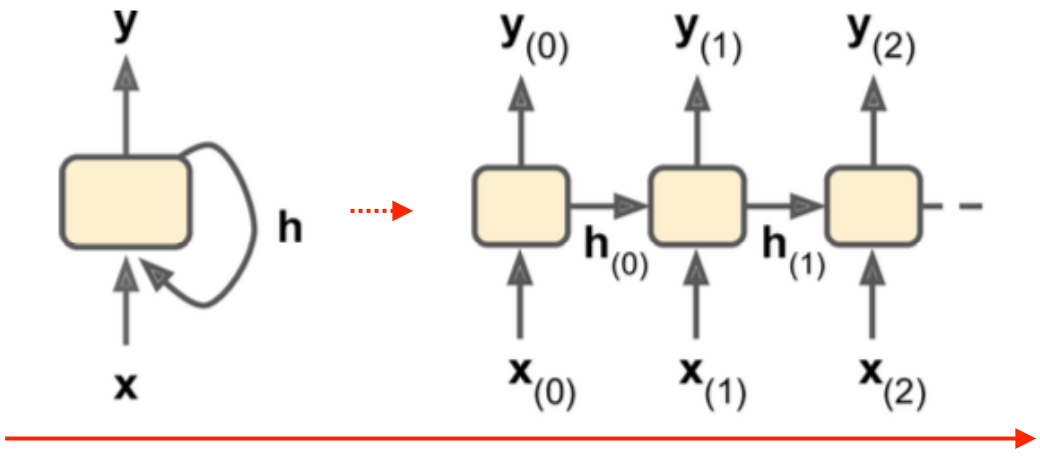


Figura de: Aurélien Gerón (2019)

RNNs

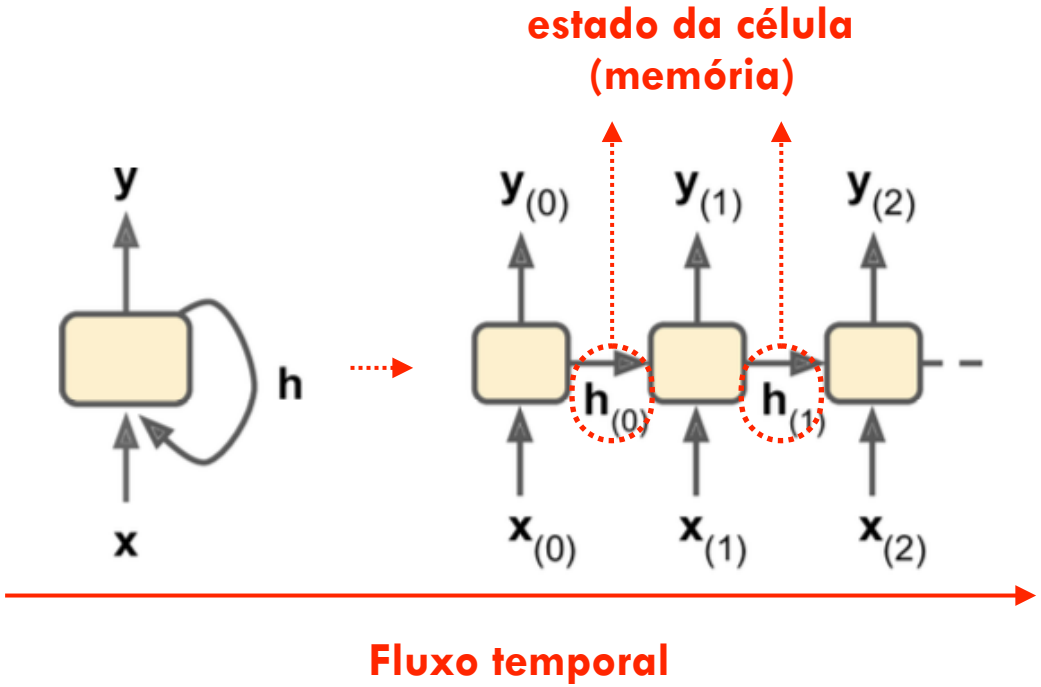


Figura de: Aurélien Gerón (2019)

Roteiro

Tipos de RNNs

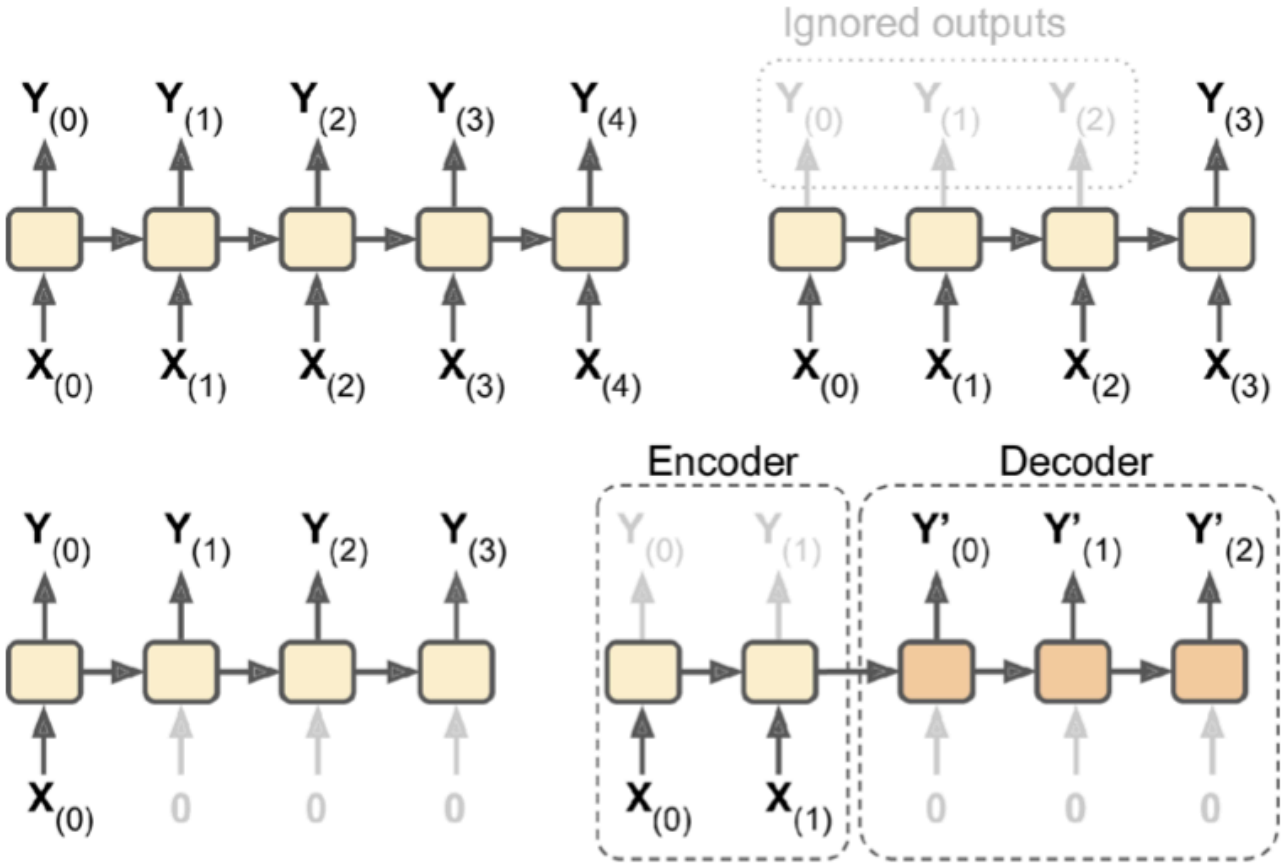
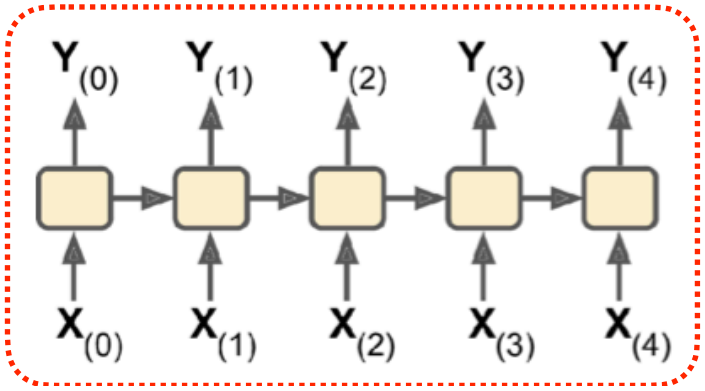


Figura de: Aurélien Gerón (2019)

Tipos de RNNs

1



01 - Sequence-to-Sequence

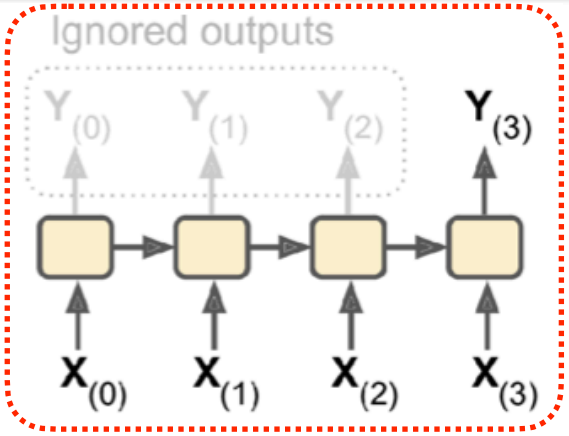
- útil para predição de séries temporais como preços de ações (*stock prices*)
- alimenta a rede com preços dos últimos N dias, e a rede prevê os preços do próximo dia (um dia a frente)

Figura de: Aurélien Gerón (2019)

Tipos de RNNs

02 - Sequence-to-vector

- ignora todas as saídas exceto a última
- alimentar a rede com uma sequência de palavras (review de um filme)
- output é um score de sentimento, de [-1, +1]:
  - -1 (ódio)
  - +1 (amor)

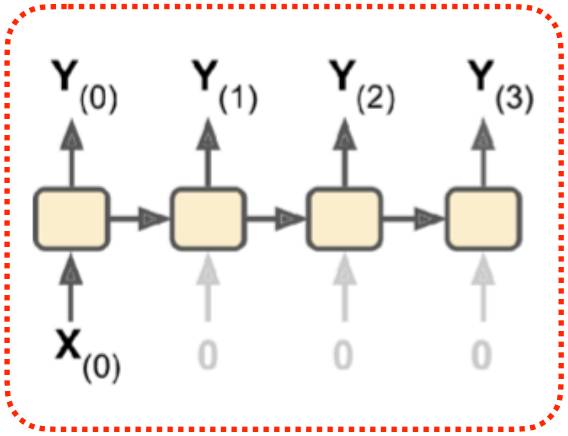


2

Figura de: Aurélien Gerón (2019)

Tipos de RNNs

3



03 - Vector-to-Sequence

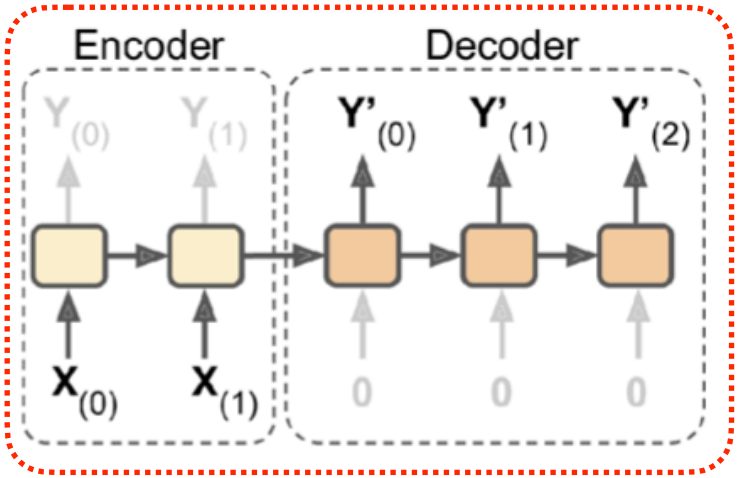
- alimenta a rede com o mesmo exemplo de entrada (x) todos os instantes de tempo, até obter uma resposta final
- Exemplo: input é uma imagem, e o output pode ser um título para a imagem

Figura de: Aurélien Gerón (2019)

Tipos de RNNs

04 - Encoder-Decoder

- *Sequence-to-Vector* → *Vetor-to-Sequence*
- Pode ser usada para gerar a tradução de uma frase entre diferentes linguagens
- processamento de linguagem natural



4

Figura de: Aurélien Gerón (2019)

Roteiro

1. Introdução
2. Redes Neurais Recorrentes (*RNNs*)
3. Tipos de RNNs
4. **Treinamento de RNNs**
5. *LSTMs*
6. Síntese / Próximas Aulas
7. Referências

Treinamento

- Treinamento:
  - **trick**: estender a rede ao longo do tempo (t)
  - aplicar backpropagation
  - BPTT: *backpropagation through time*

Treinamento

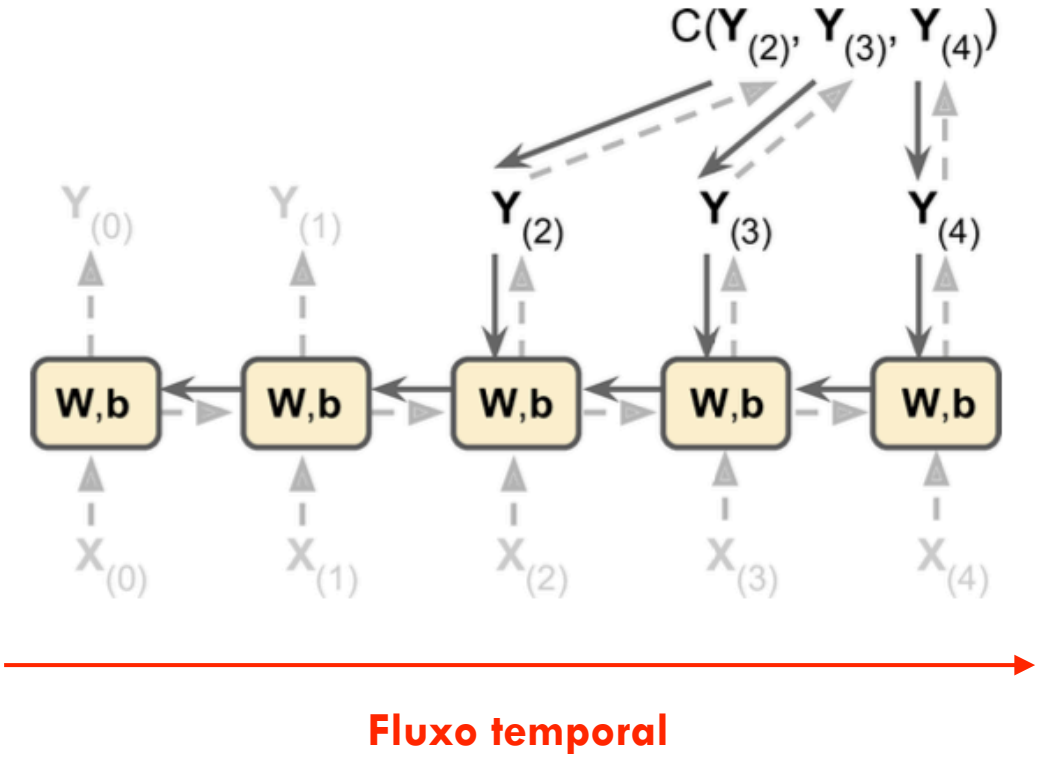


Figura de: Aurélien Gerón (2019)

Treinamento

1

sinal flui pela rede (forward)

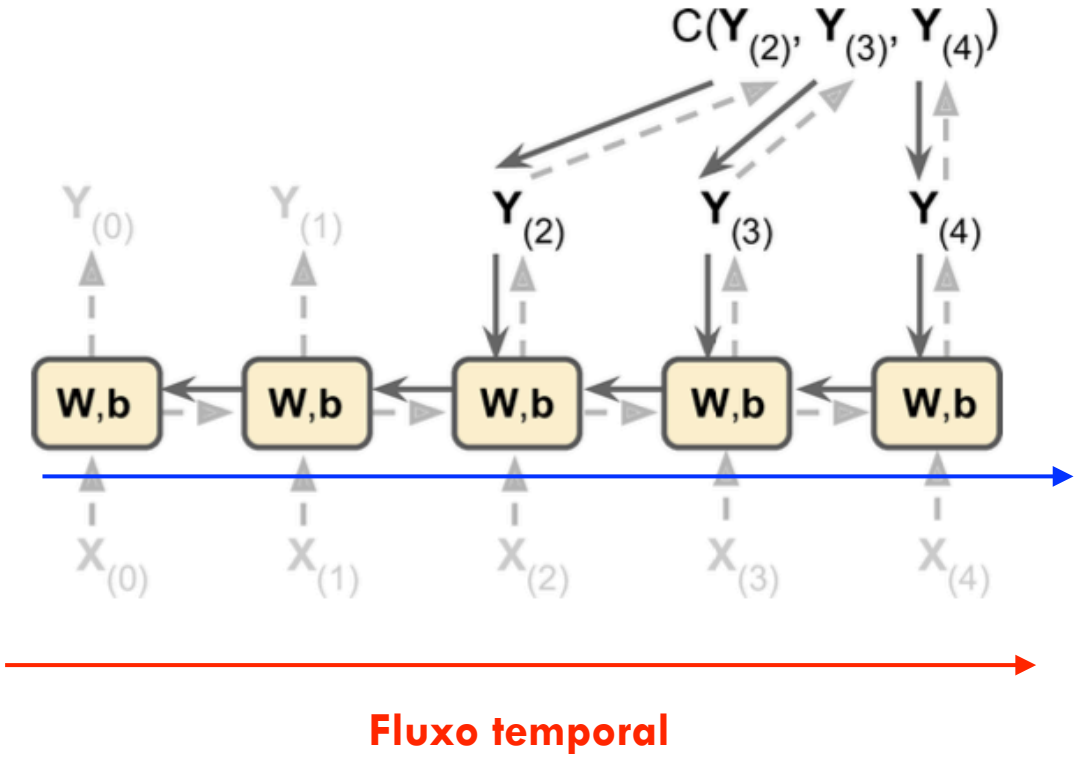


Figura de: Aurélien Gerón (2019)

Treinamento

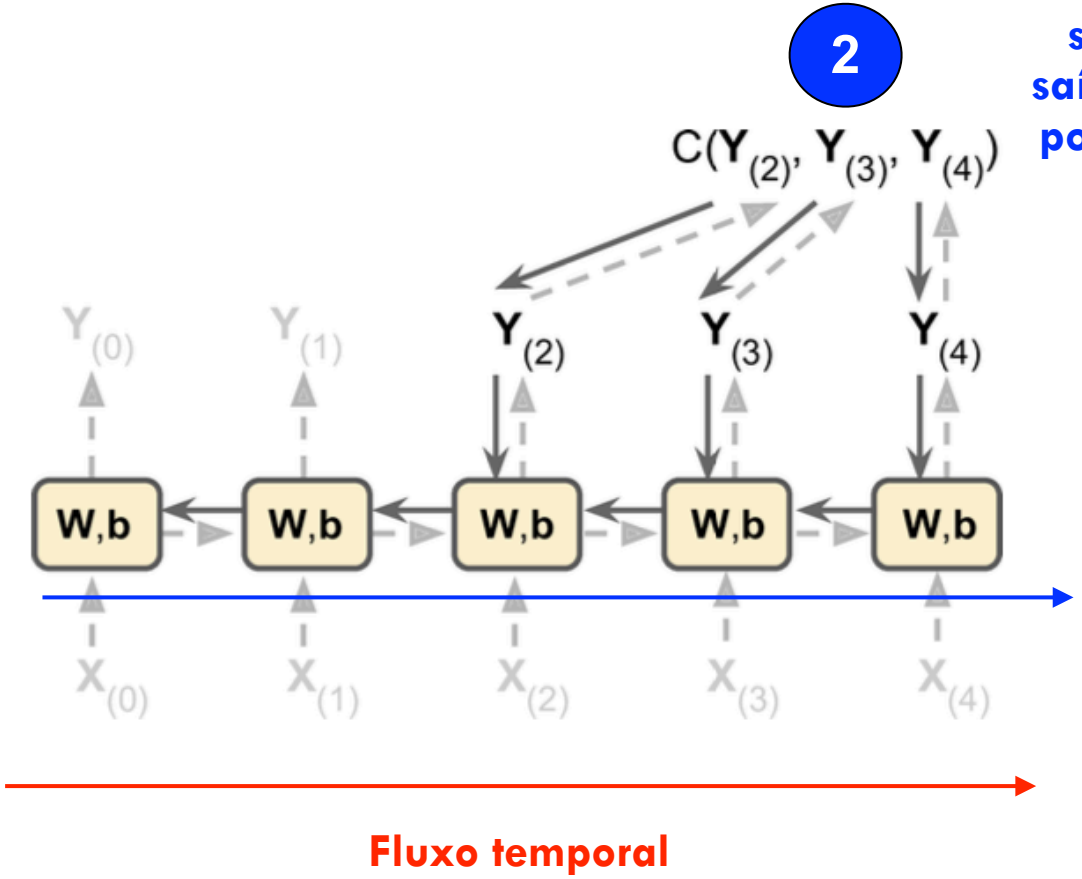


Figura de: Aurélien Gerón (2019)

Treinamento

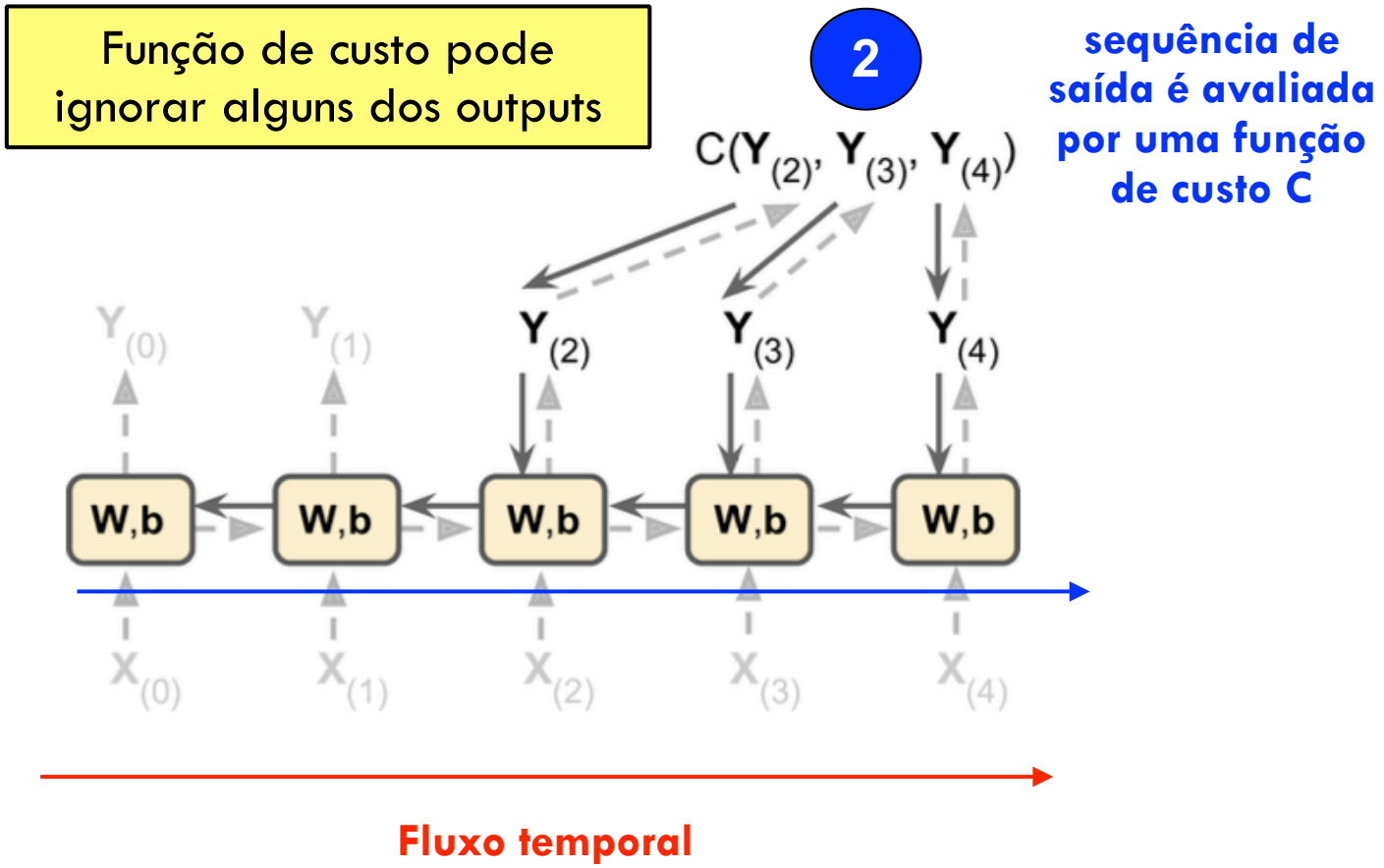


Figura de: Aurélien Gerón (2019)

Treinamento

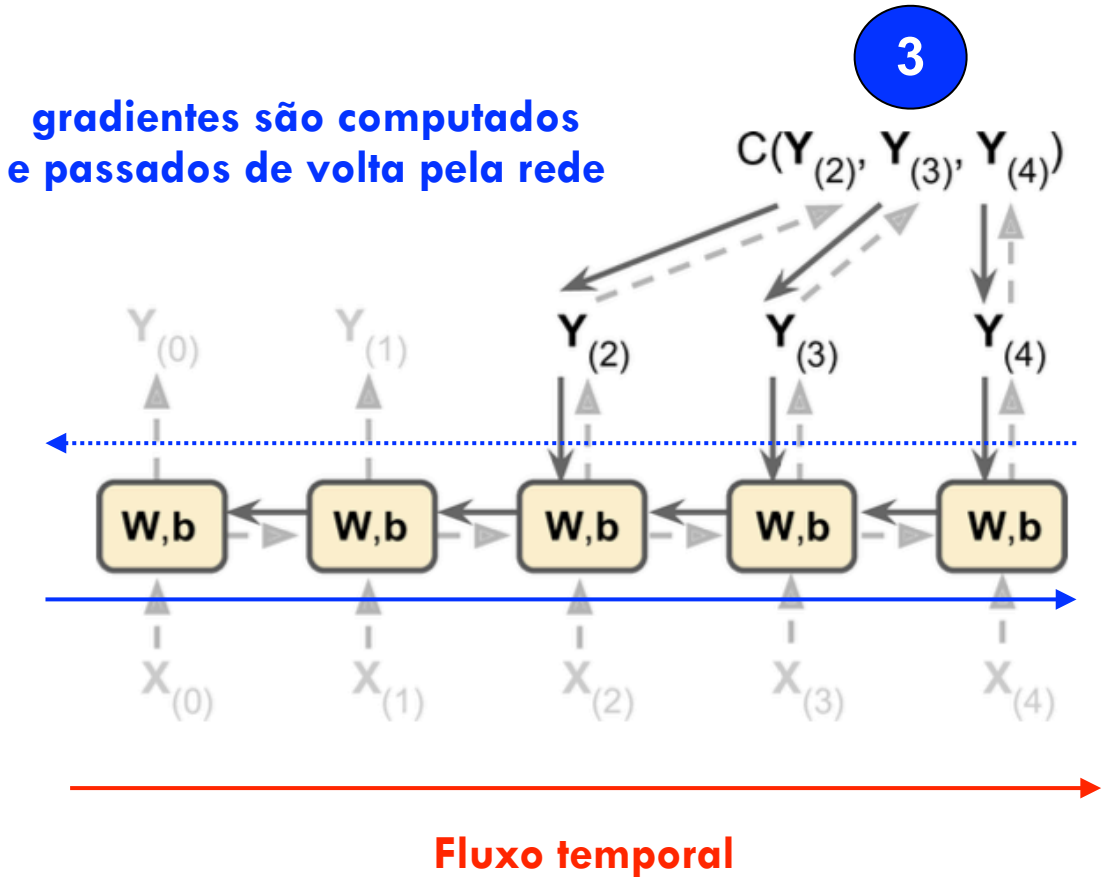


Figura de: Aurélien Gerón (2019)

Treinamento

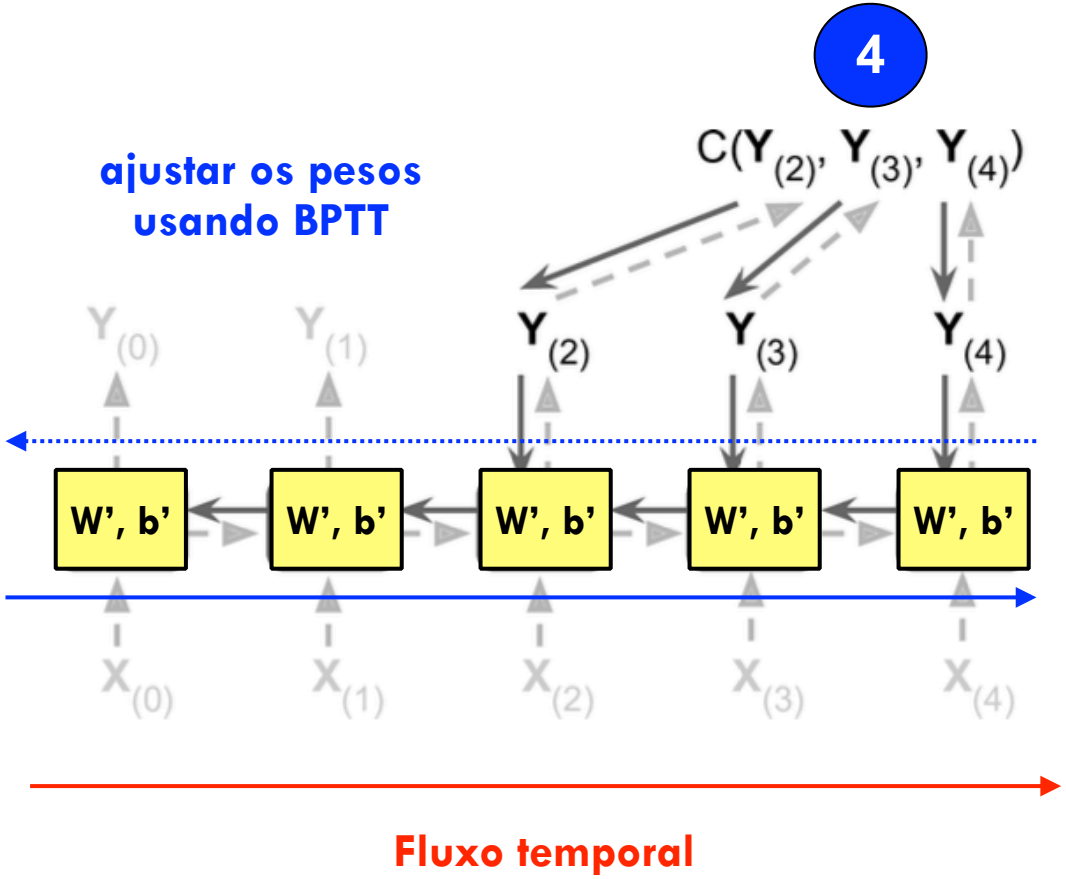


Figura de: Aurélien Gerón (2019)

Treinamento

- Passos gerais para treinamento de RNNs:
  - 1. Propagar sinal
  - 2. Computar a estimativa de erro (custo)
  - 3. Calcular os gradientes
    - propagar de volta pela rede, considerando todos os instantes de tempo
  - 4. Atualizar os pesos (W, b) via BPTT

Treinamento

Obs: os gradientes voltam e consideram **todas as saídas** (*outputs*) usadas pela função de custo, não apenas o último valor

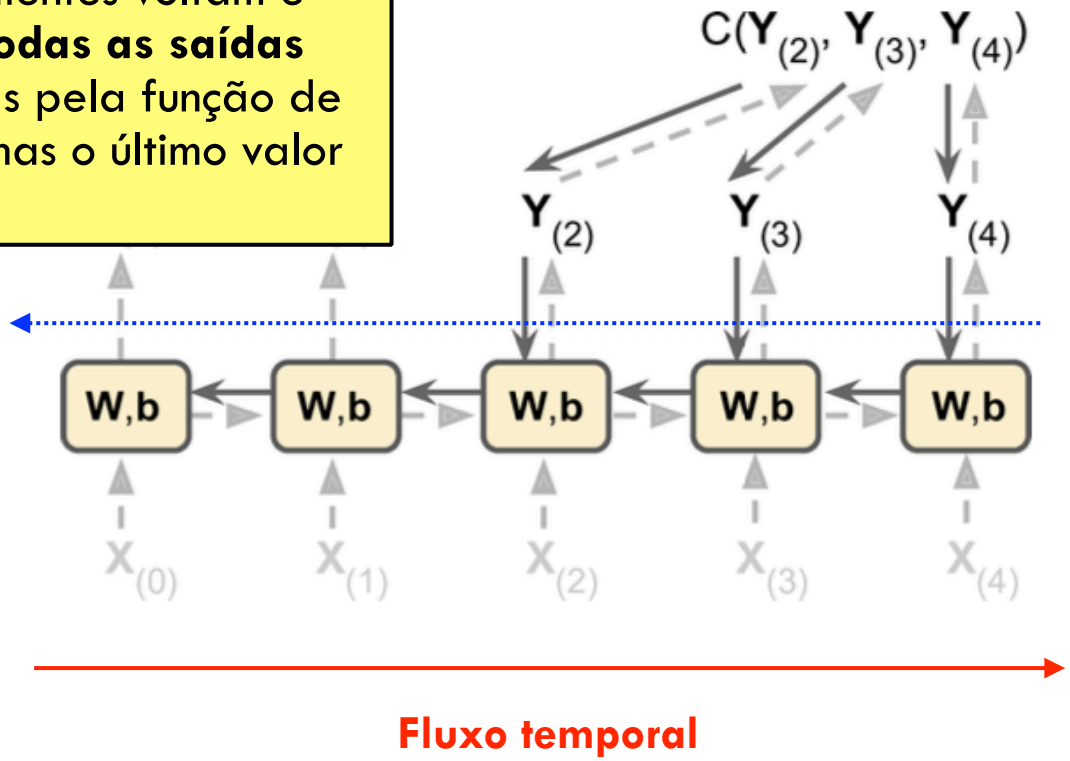


Figura de: Aurélien Gerón (2019)

Treinamento

Obs 2: como os mesmos parâmetros (W, b) são usados a cada instante de tempo, o BPTT faz o **ajuste** considerando **todos os instantes de tempo**

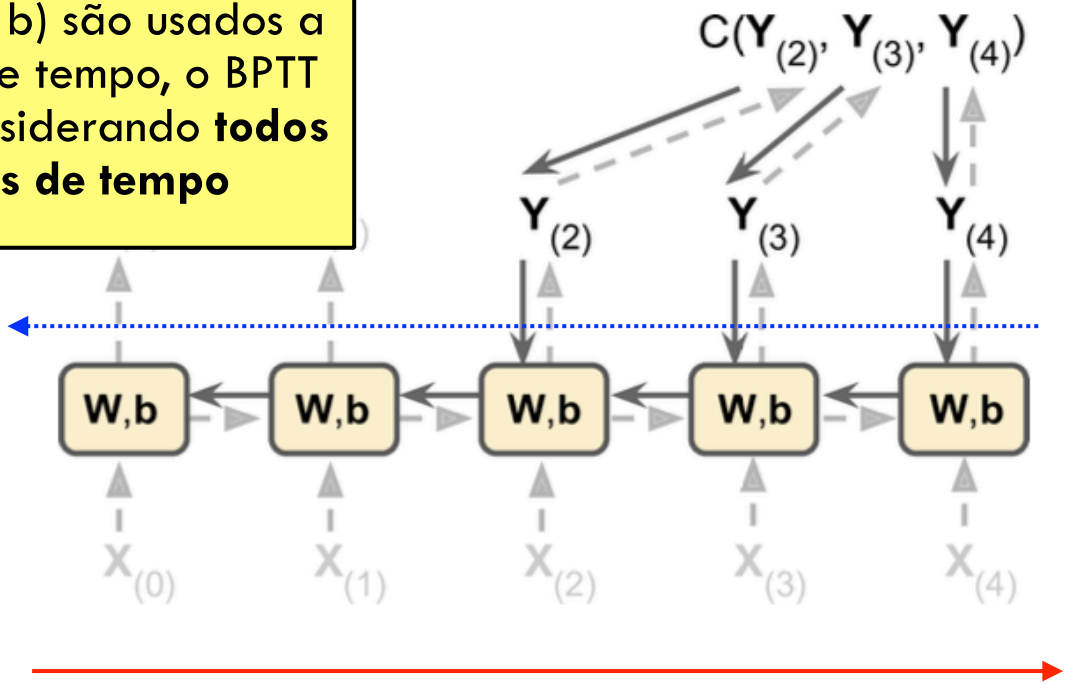


Figura de: Aurélien Gerón (2019)

Roteiro

1. Introdução
2. Redes Neurais Recorrentes (*RNNs*)
3. Tipos de RNNs
4. Treinamento de RNNs
5. *LSTMs*
6. Síntese / Próximas Aulas
7. Referências

LSTMs

- Quando uma RNN é treinada em um grande conjunto/sequência de dados, a rede executa por muitos instantes de tempo. Logo:
  - a **rede estendida** ao longo do tempo fica **muito profunda**
  - pode sofrer de **gradientes instáveis**
  - gradualmente vai começar a **esquecer padrões** ou informações importantes das **primeiras entradas**

LSTMs

- Soluções:
  - O que já funciona com os outros modelos de DL:
    - melhor inicialização dos pesos
    - dropout, normalização, otimizadores, etc …
  - Unidades/Neurônios mais complexos
    - exemplo: neurônios **LSTMs**
  - O que não ajuda?
    - ReLU → estoura o valor das ativações
      - usar $\phi = \tanh(\ )$ [**default**]

LSTMs

- Proposto por:
  - Sepp Hochreiter & Jurgen Schmidhuber (1997)

- Gradualmente melhorada:
  - Alex Graves (2013)
  - Hasim Sak et al (2014)
  - Wojciech Zaremba et al (2014)

LSTMs

Como um neurônio/célula/unidade LSTM funciona?

LSTMs

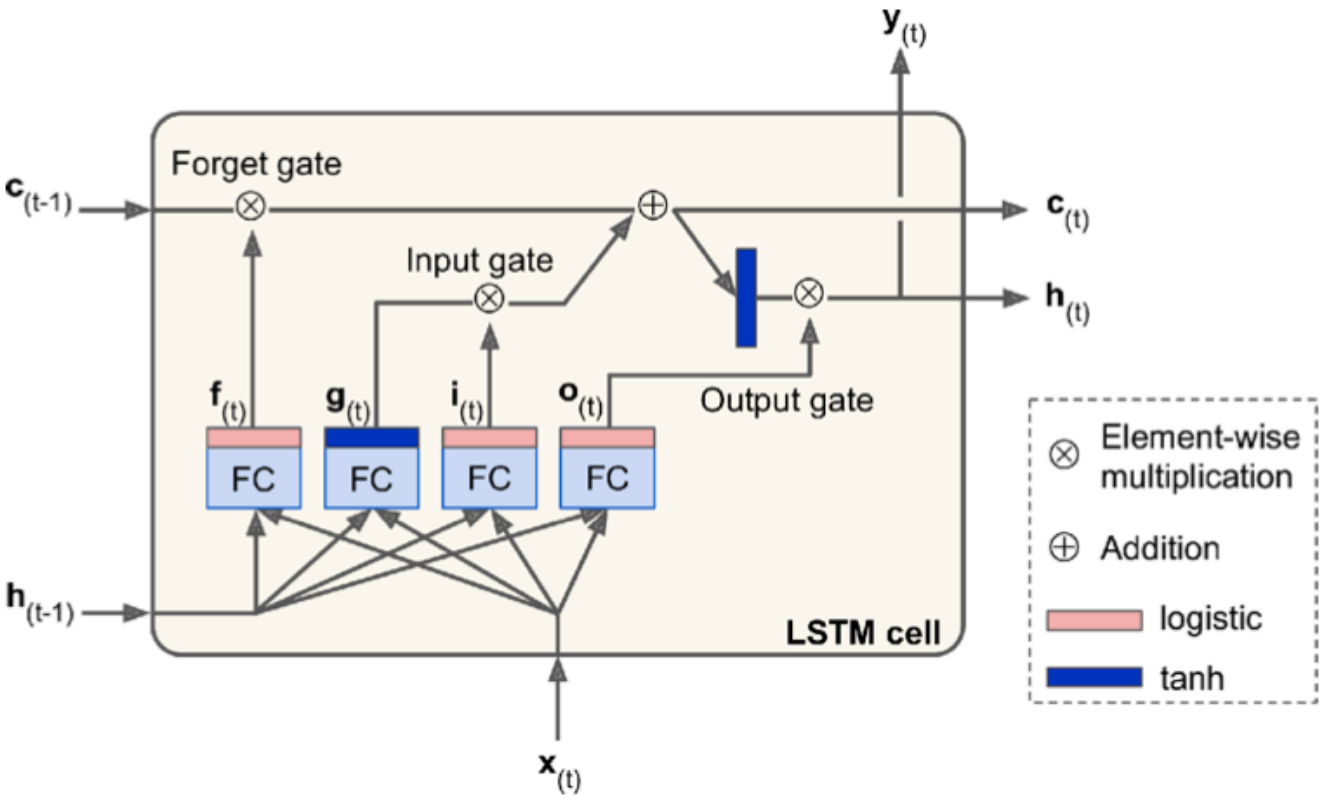


Figura de: Aurélien Gerón (2019)

LSTMs

Ideia: rede pode aprender um estado/padrão de longa duração, descartar o que não interessa, e o manter o que pode ser interessante

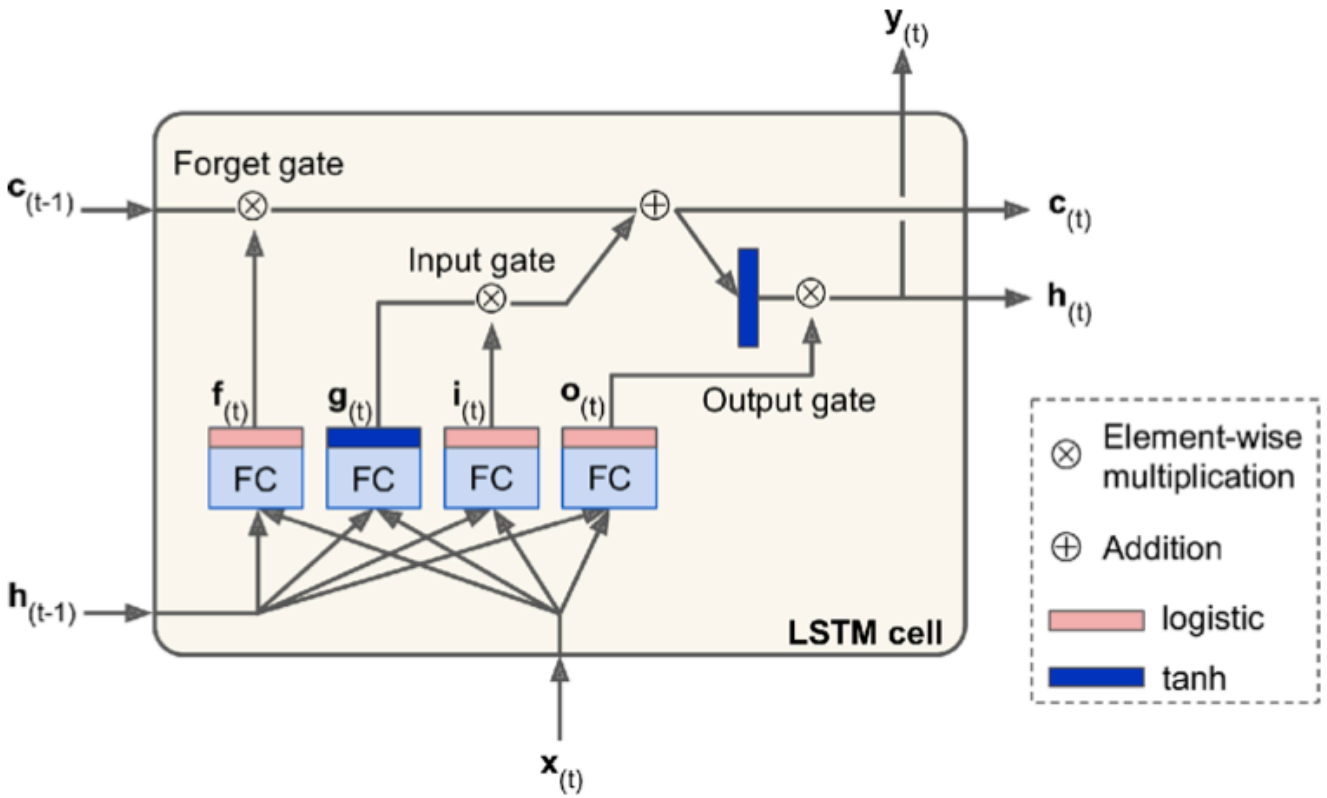


Figura de: Aurélien Gerón (2019)

LSTMs

- **Long-term (c):**
  - c(t-1) flui pela rede da esquerda para a direita, passa primeiro por um portão de "esquecimento" (forget gate), descarta algumas memórias (dados)
  - e depois adiciona mais memórias via operação de adição dos sinais
  - adiciona as memórias selecionadas pelo portão de entrada (*input gate*)
  - o resultado c(t) é passado adiante, sem nenhuma transformação

LSTMs

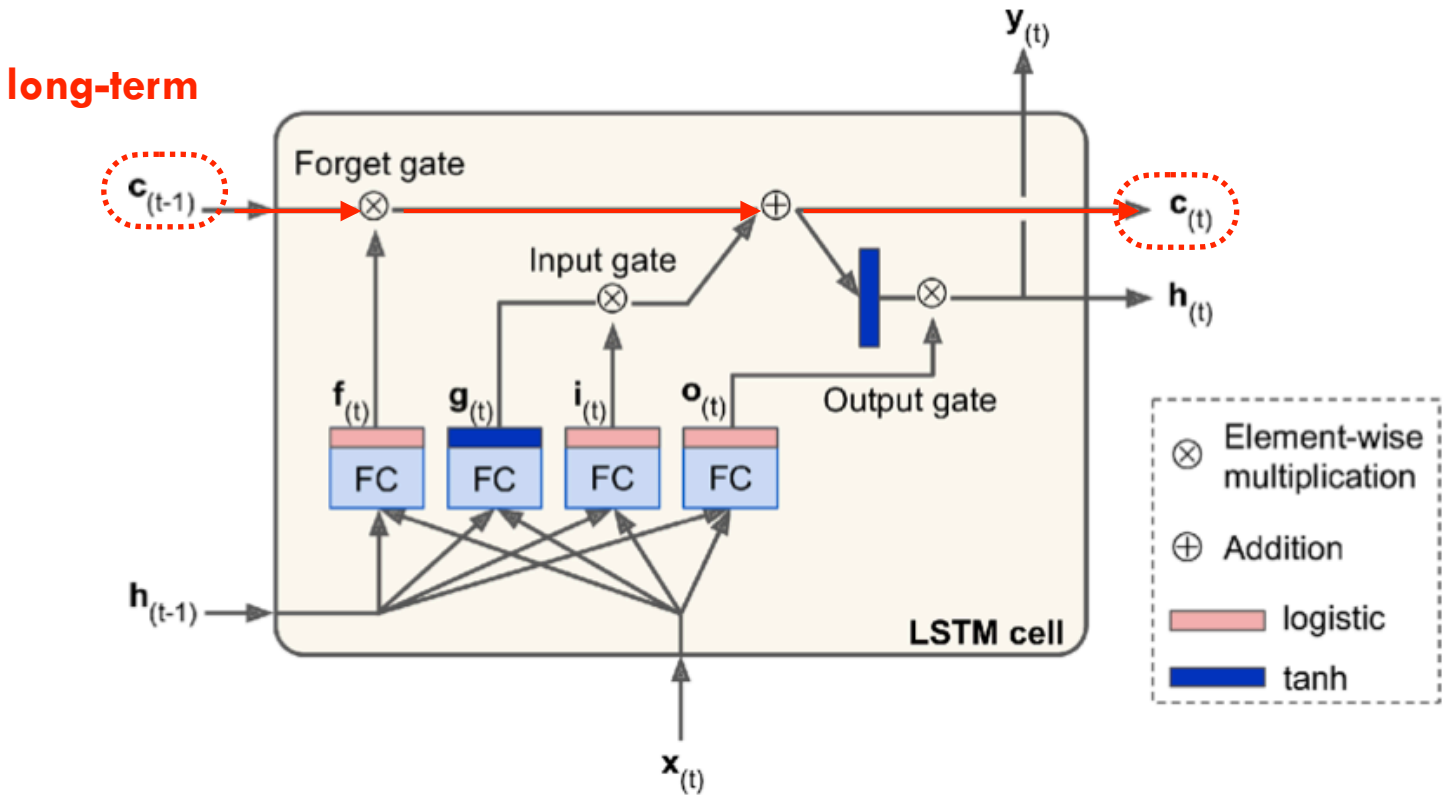


Figura de: Aurélien Gerón (2019)

LSTMs

- A cada instante de tempo (t):
  - memórias são esquecidas
  - memórias são adicionadas/reforçadas

- Depois da soma, o estado de **longo-termo** (*long-term*) (c) é copiado e passado para a função de ativação (tanh)
  - o resultado é filtrado pelo portão de saída (*output gate*)

- Esse processo resultado no **estado de curto-termo** (*short-term*) h(t), que é **igual à saída** do neurônio para t, **y(t)**

LSTMs

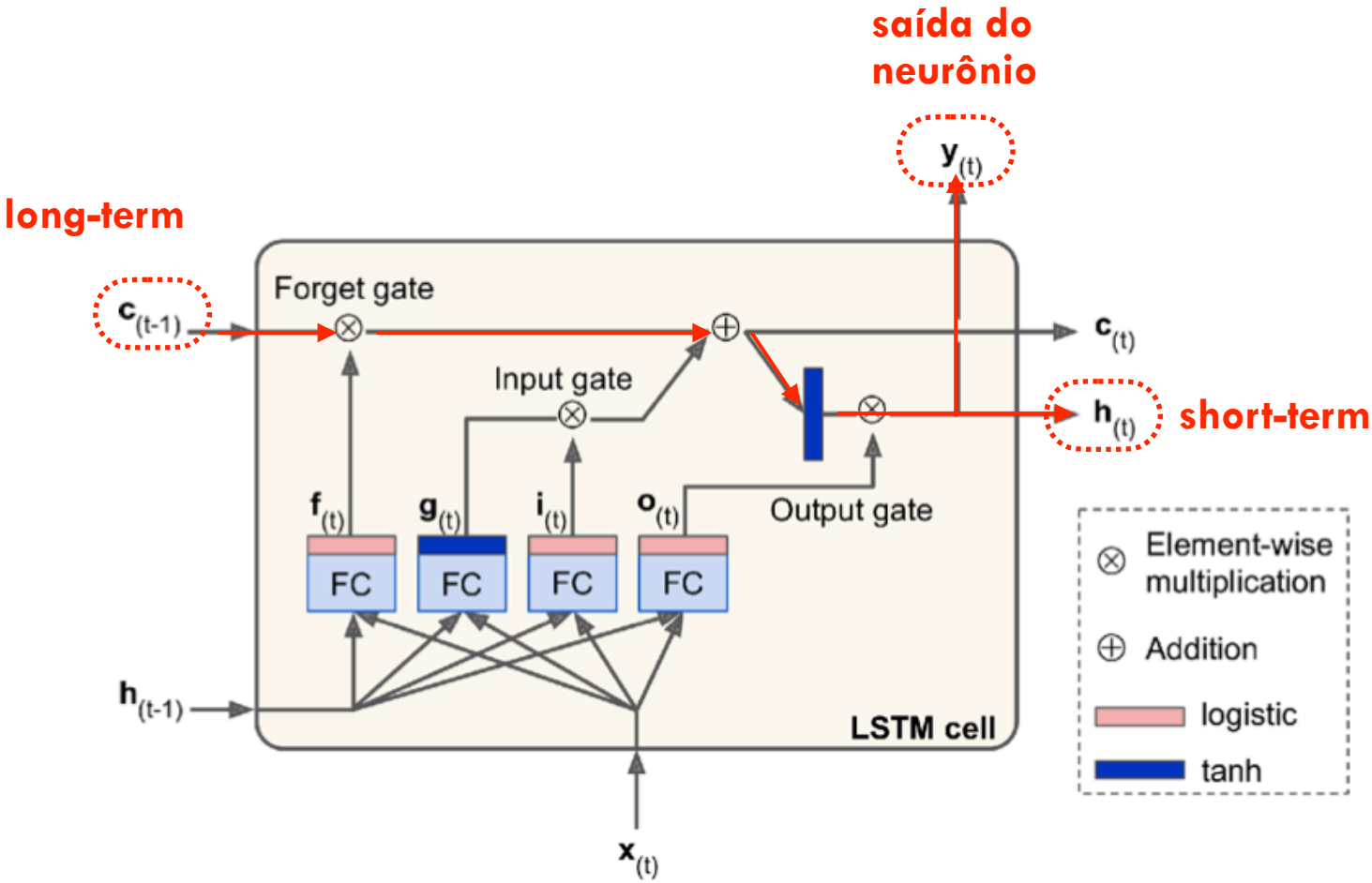


Figura de: Aurélien Gerón (2019)

LSTMs

- De onde vem o processo de memorização?

LSTMs

- De onde vem o processo de memorização?
  1. Entrada atual **x(t)** e o short-term anterior h(t-1) são alimentados a quatro camadas totalmente conectadas (FC)

LSTMs

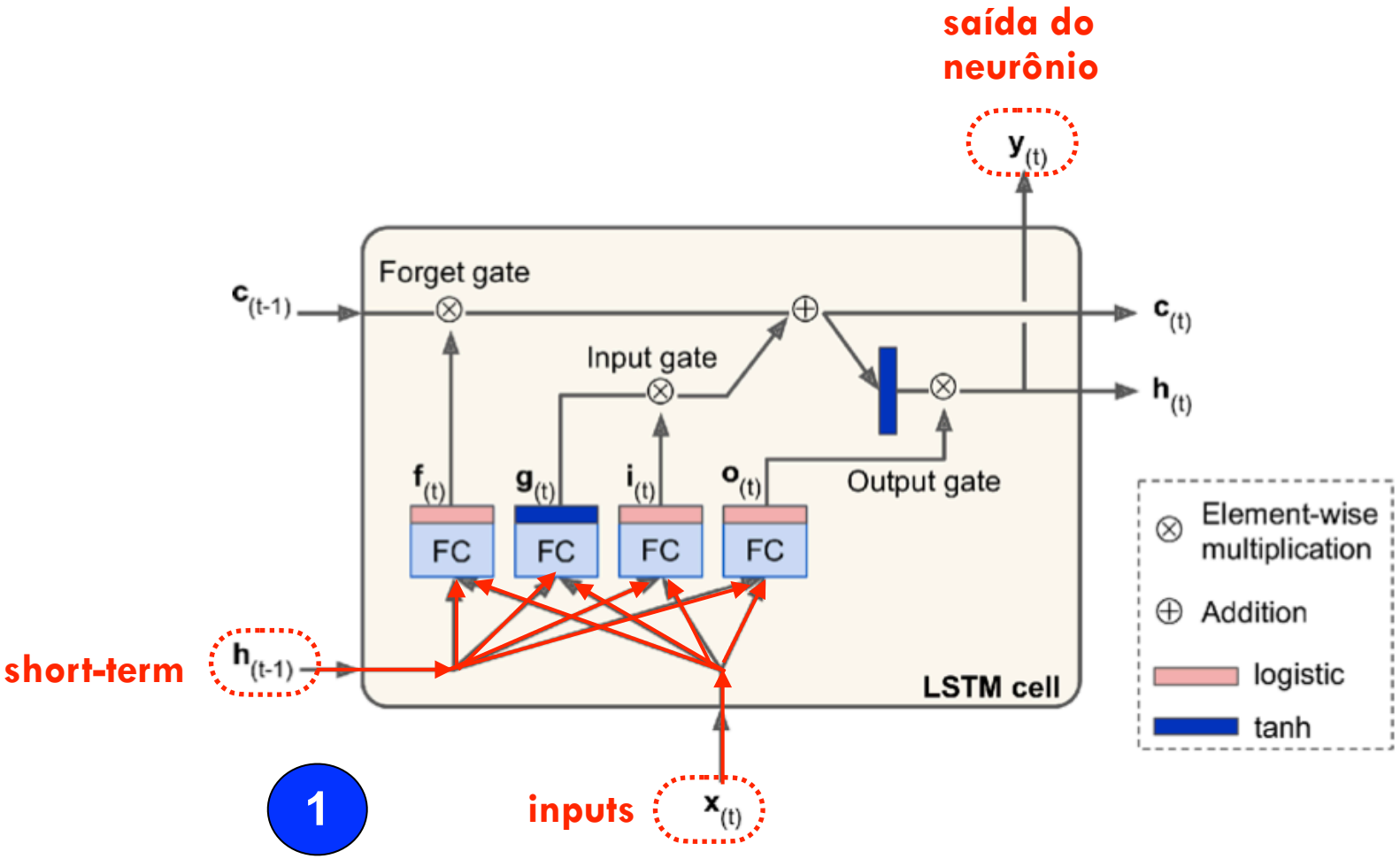


Figura de: Aurélien Gerón (2019)

LSTMs

- De onde vem o processo de memorização?
  1. Entrada atual **x(t)** e o short-term anterior h(t-1) são alimentados a quatro camadas totalmente conectadas (FC)
  2. A principal camada FC é a que gera o sinal **g(t)**
     - papel é analizar as entradas atuais x(t) e o estado de curto prazo (*short-term*) anterior **h(t-1)**
     - maior parte da saída dessa camada é armazenada no estado de longo prazo (*long-term*), o resto é esquecido

LSTMs

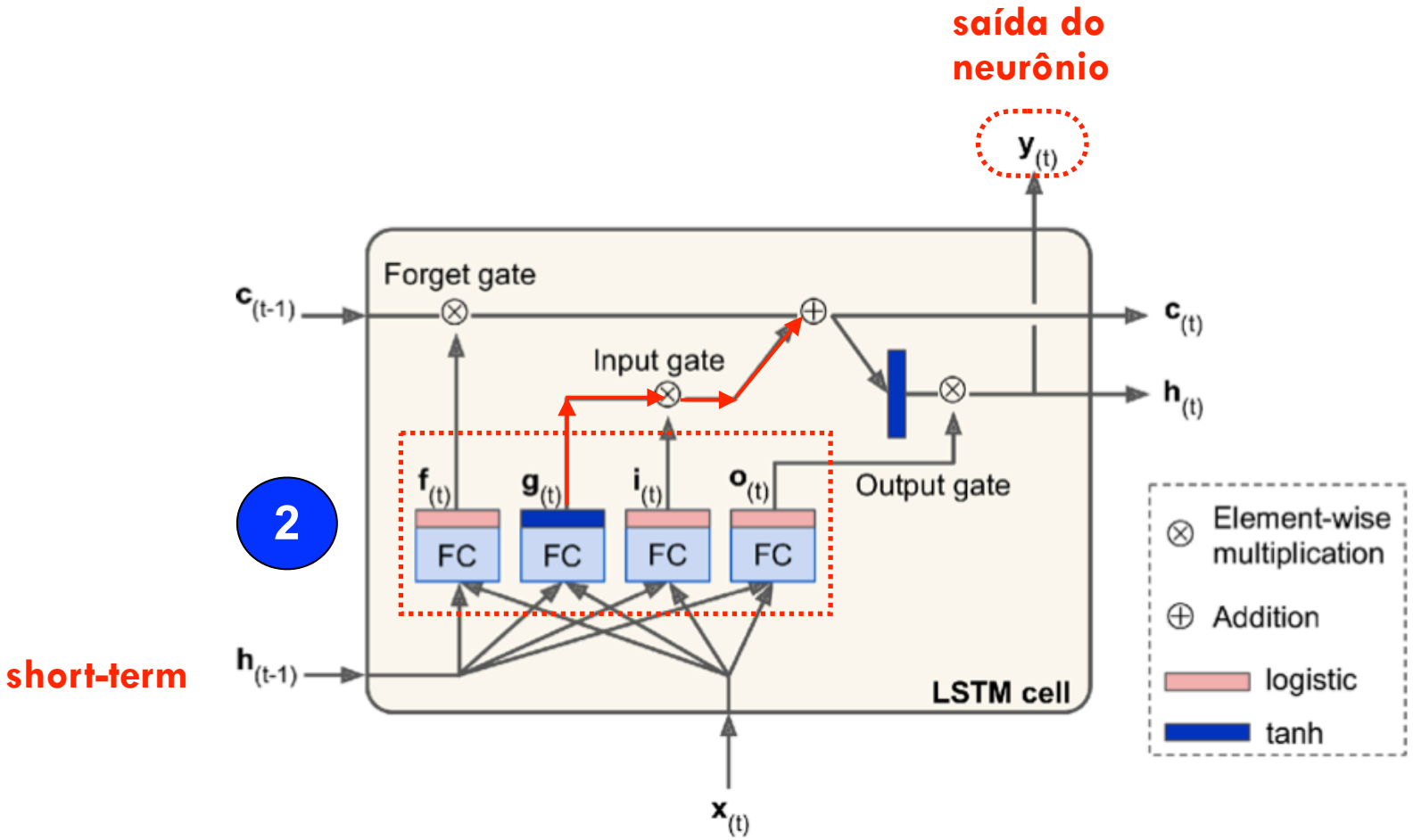


Figura de: Aurélien Gerón (2019)

LSTMs

- De onde vem o processo de memorização?
  1. Entrada atual **x(t)** e o short-term anterior h(t-1) são alimentados a quatro camadas totalmente conectadas (FC)
  2. A principal camada FC é a que gera o sinal g(t)
  3. As outras três camadas são portões de controle (*gates*)
     - possuem ativação logística
     - saídas variam de 0 a 1
     - outputs alimentam operação de multiplicação elemento a elemento (*element-wise*)
     - 0s → portões fechados, 1s → portões abertos

LSTMs

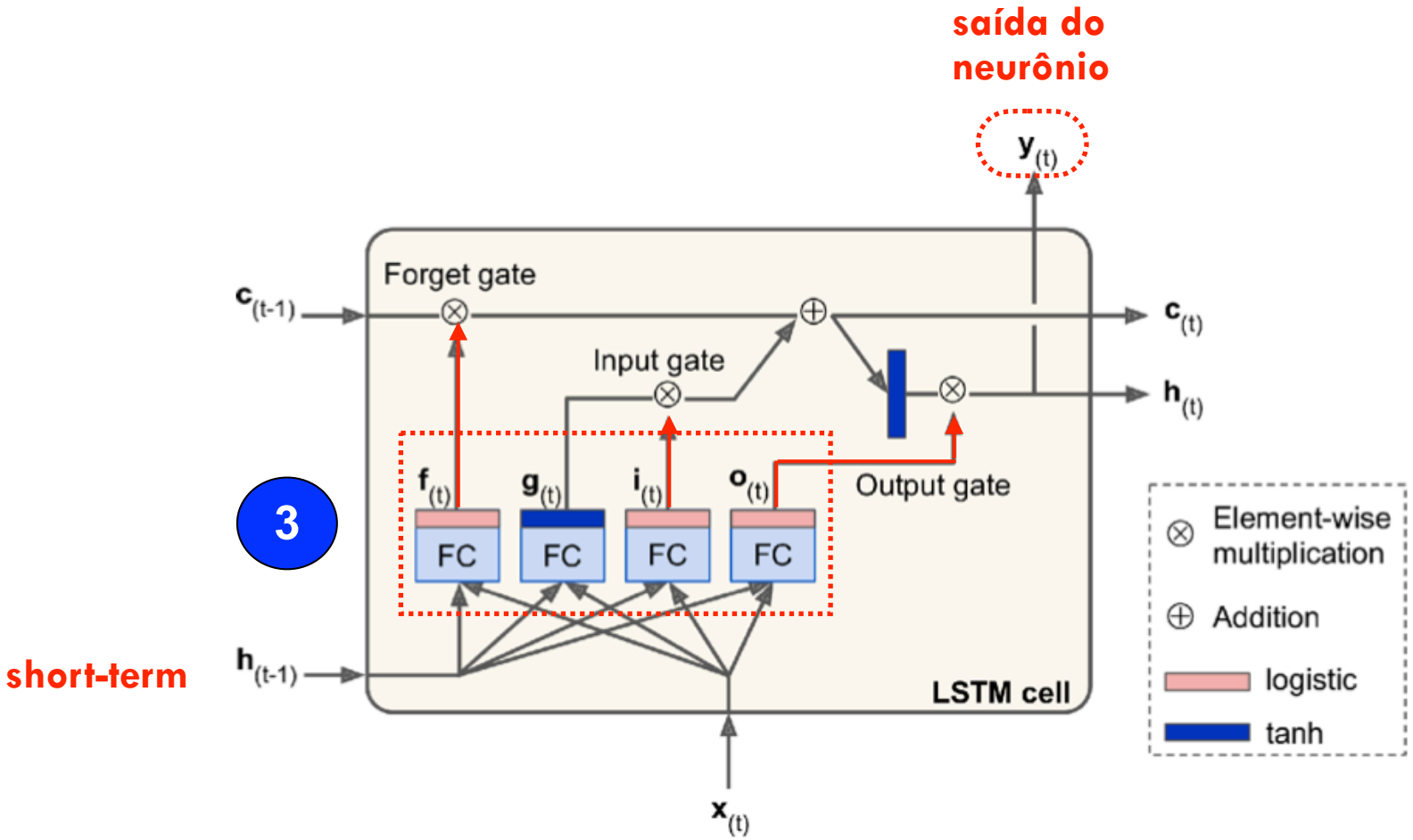


Figura de: Aurélien Gerón (2019)

LSTMs

Operação *element-wise* dos portões (*gates*):
operação posição por posição

$$a \circ b = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}_{(n \times 1)} \circ \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}_{(n \times 1)} = \begin{bmatrix} a_1 b_1 \\ a_2 b_2 \\ a_3 b_3 \\ a_4 b_4 \\ a_5 b_5 \end{bmatrix}_{(n \times 1)}$$

Element wise Product

LSTMs

- De onde vem o processo de memorização?
  1. Entrada atual **x(t)** e o short-term anterior h(t-1) são alimentados a quatro camadas totalmente conectadas (FC)
  2. A principal camada FC é a que gera o sinal g(t)
  3. As outras três camadas são portões de controle (*gates*)
  4. portões de controle
     - **portão de esquecimento,** controlado por **f(t),** define qual parte do estado de longo termo deve ser esquecido
     - **portão de entrada i(t),** controla qual parte de g(t) deve ser adicionado ao estado de longo-termo
     - **portão de saída, o(t),** controla qual parte do estado de longo-termo deve ser lido, e gera a saída do tempo corrente, enviando para h(t) e y(t)

LSTMs

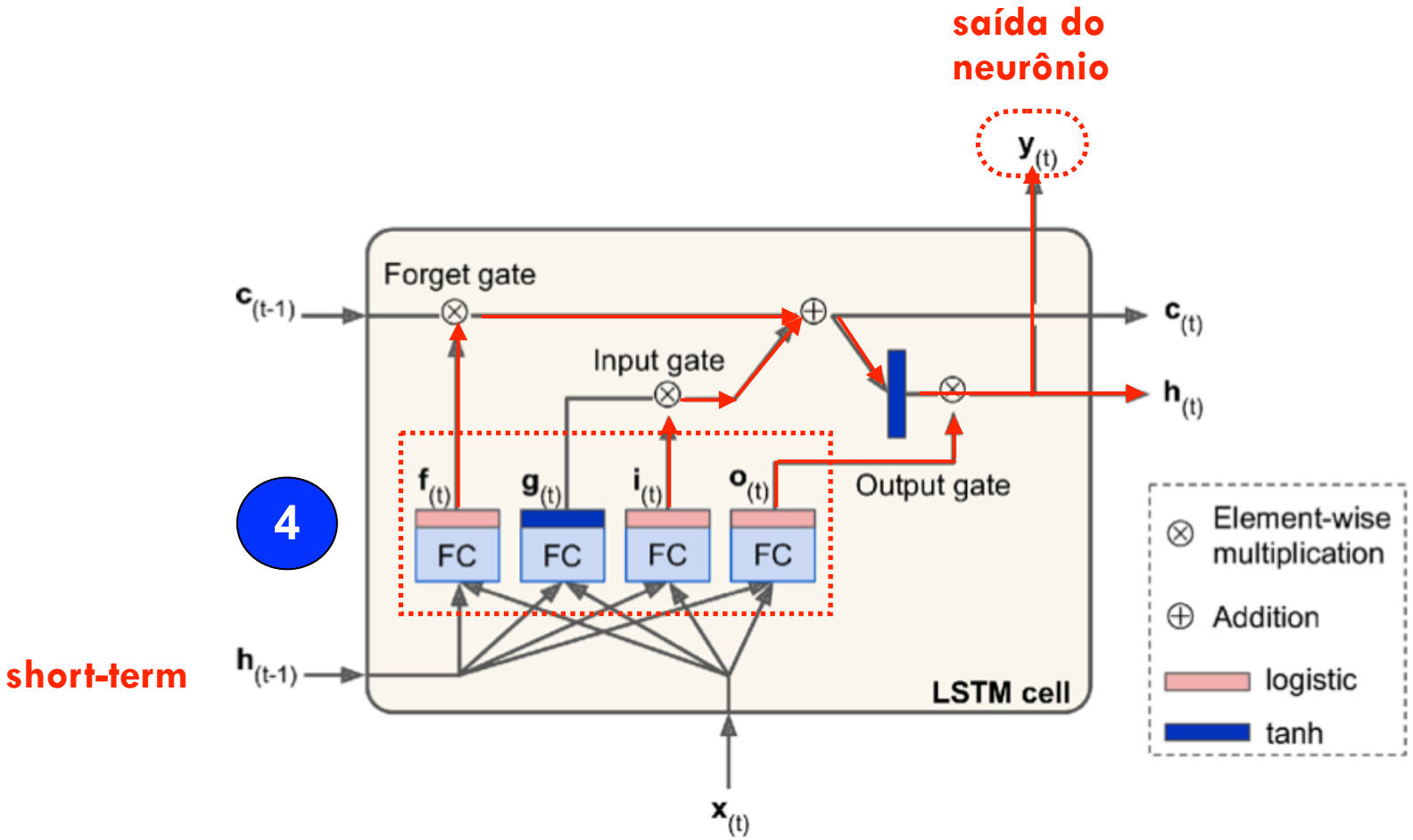


Figura de: Aurélien Gerón (2019)

LSTMs

- **Resumindo**
  - LSTM pode aprender a reconhecer entradas importantes (*input gate*)
  - *forget gate:* decide qual informação será esquecida pelo bloco
  - *input gate*: define qual informação será atualizada no state
  - *output gate*: decide qual informação será a saída do bloco

- Permite:
  - encontrar padrões longos em séries temporais (textos, áudios, etc)

LSTMs

- Equações para computar o estado/saída do neurônio LSTM:

$$i_{(t)} = \sigma(W_{xi}^T x_{(t)} + W_{hi}^T h_{(t-1)} + b_i)$$

$$f_{(t)} = \sigma(W_{xf}^T x_{(t)} + W_{hf}^T h_{(t-1)} + b_f)$$

$$o_{(t)} = \sigma(W_{xo}^T x_{(t)} + W_{ho}^T h_{(t-1)} + b_o)$$

$$f_{(t)} = tanh(W_{xg}^T x_{(t)} + W_{hg}^T h_{(t-1)} + b_g)$$

$$c_{(t)} = f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)}$$

$$y_{(t)} = h_{(t)} = o_{(t)} \otimes tanh(c_{(t)})$$

LSTMs

- onde:
  - σ: função logística
  - tanh: função tangente hiperbólica
  - *Wxi, Wxf, Wxo, Wxg* são as matrizes de peso conectando cada camada interna com o vetor de entradas X(t)
  - *Whi, Whf, Who, Who* são as matrizes conectando cada camada com o estado de curto termo do estado anterior (short-term) **h(t-1)**
  - *bi, bf, bo* and *bg* são os bias de cada camada
    - bf is inicializado como 1 ao invés de 0. Prevenir esquecer as coisas no começo do treinamento

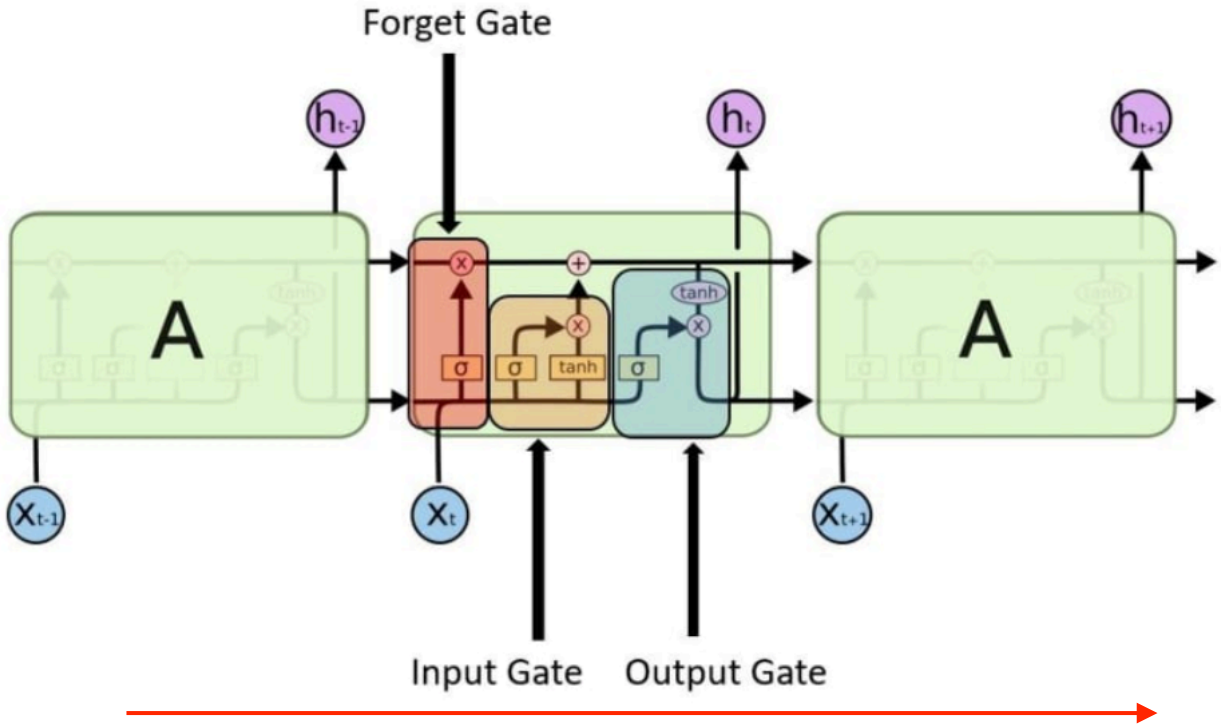
LSTMs

pseudocódigo LSTMCell
Inptus: prev_ct, prev_ht, input #estados anteriores e input
Outputs: ht, ct, yt

1. combine = prev_ht + input #combina hidden state com input
2 ft = forget_layer(combine) # aplicando esquecimento na comb.
3. candidate = candidate_layer(combine) # ajusta comb.
4. it = input_layer(combine) # it é o ajuste da combinação
5. ct = prev_ct * ft + candidate # atualiza cell state (long-term)
6. ot = output_layer(combine) # encontrando output
7. yt = ht = ot * tanh(ct) # calculando saída da cell
8. return ht, yt, ct

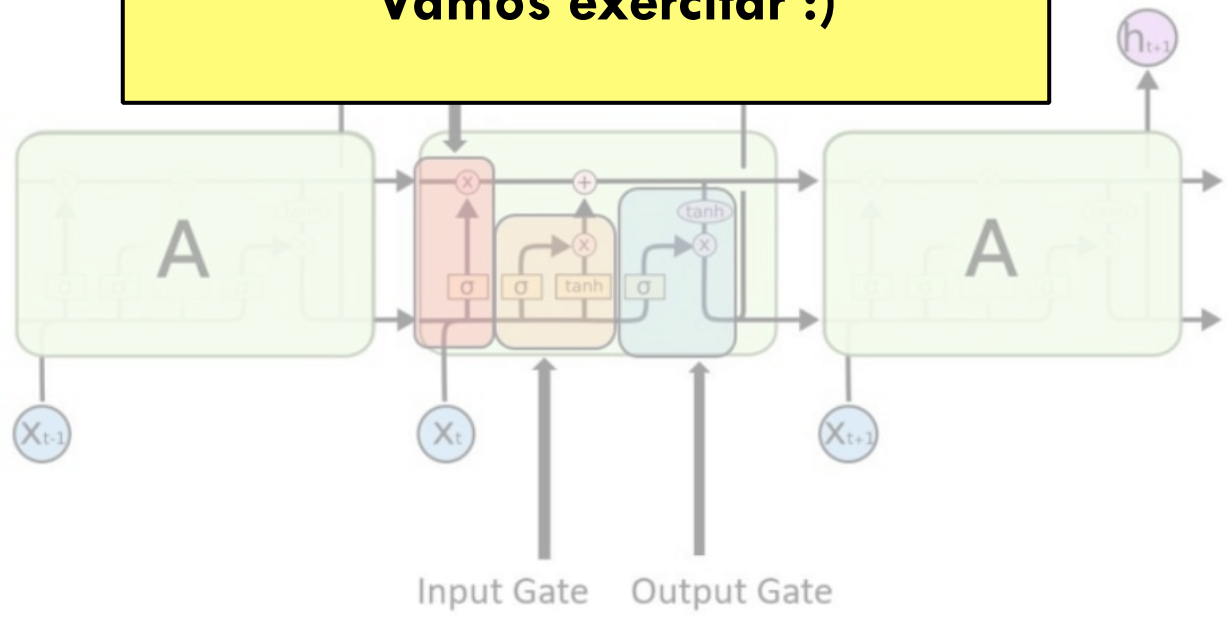
LSTMs

Lembrar que a **LSTM** se estende ao **longo do tempo …**



Hands on

Vamos exercitar :)



Roteiro

1. Introdução
2. Redes Neurais Recorrentes (*RNNs*)
3. Tipos de RNNs
4. Treinamento de RNNs
5. *LSTMs*
6. Síntese / Próximas Aulas
7. Referências

Síntese

- **RNNs**
  - usadas para predição de dados temporais
  - rede se estende ao longo do tempo
  - treinamento via BPTT
  - problemas (gradiente, memória curta)
  - LSTMs

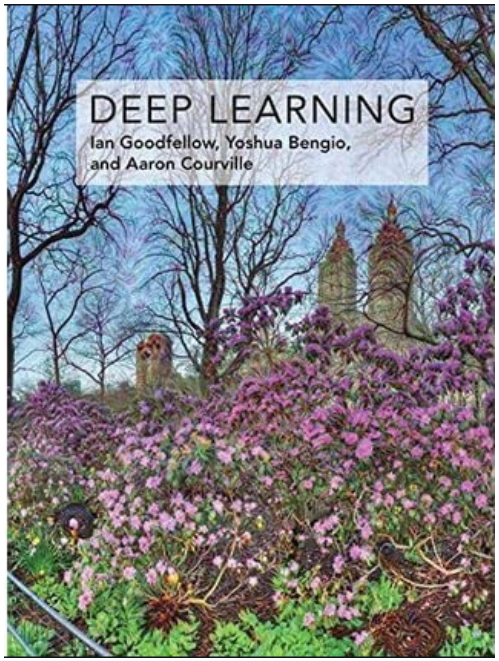
Próxima aula

- *Autoencoders*
  - visão geral
  - compressão de informação
  - etc …

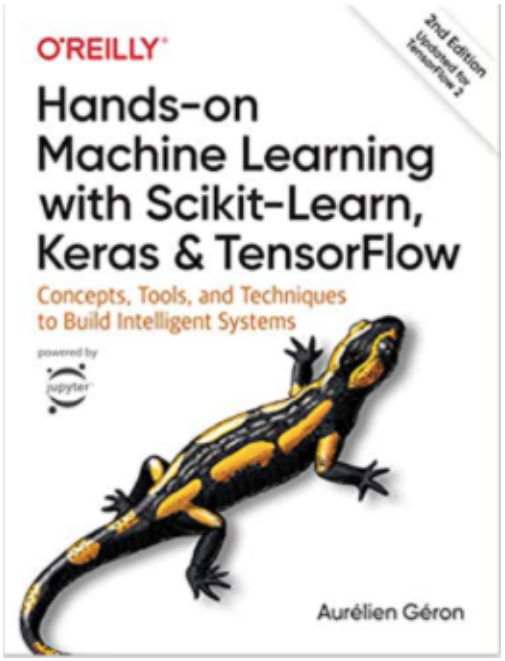
Roteiro

1. Introdução
2. Redes Neurais Recorrentes (*RNNs*)
3. Tipos de RNNs
4. Treinamento de RNNs
5. *LSTMs*
6. Síntese / Próximas Aulas
7. Referências

Literatura Sugerida



(Goodfelow, Bengio, Courville; 2015)



(Géron, 2019)

Artigos

- Sepp Hochreiter and Jürgen Schmidhuber, “Long Short-Term Memory,” *Neural Computation* 9, no. 8 (1997): 1735–1780.
- Alex Graves, “Generating Sequences with Recurrent Neural Networks”, arxiv preprint arXiv:1308.0850 (2013).
- HaŞim Sak et al., “Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition,” arXiv preprint arXiv:1402.1128 (2014).
- Wojciech Zaremba et al., “Recurrent Neural Network Regularization,” arXiv preprint arXiv:1409.2329 (2014).

Literatura Complementar

- MIT book: http://www.deeplearningbook.org
- Deep Learning: http://deeplearning.net
- Andrew Ng: https://www.deeplearning.ai

Literatura Complementar

- Coursera: https://www.coursera.org/specializations/deep-learning
- Google AI: https://ai.google/education/
- Keras: https://keras.io
- Auto-Keras: https://autokeras.com
- h2o: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html#

Obrigado :)

Prof. Rafael G. Mantovani

rafaelmantovani@utfpr.edu.br