Bloco LSTM

PROF. CLODOALDO A M LIMA

Introdução

Para compreender devidamente redes neurais recorrentes, é necessário dominar conceitos fundamentais associados a sistemas dinâmicos.

O conceito mais básico é o de **estado do sistema**, que computacionalmente é implementado na forma de uma memória interna.

O estado pode ser representado na forma de um vetor de variáveis, capazes de caracterizar completamente a configuração atual do sistema. Mais do que isso: com o conhecimento do estado do sistema no instante presente, de sua dinâmica e das entradas que atuam sobre o sistema do instante presente em diante, o que vai ocorrer com o sistema dinâmico do instante presente em diante não depende do que ocorreu com o sistema no passado, pois tudo está consolidado nas variáveis de estado do sistema. Logo, as variáveis de estado, por definição, integram tudo o que aconteceu com o sistema no passado.

Introdução

Definição formal: O estado de um sistema é formado por um conjunto de valores de grandezas físicas, necessário e suficiente para caracterizar a situação física deste sistema.

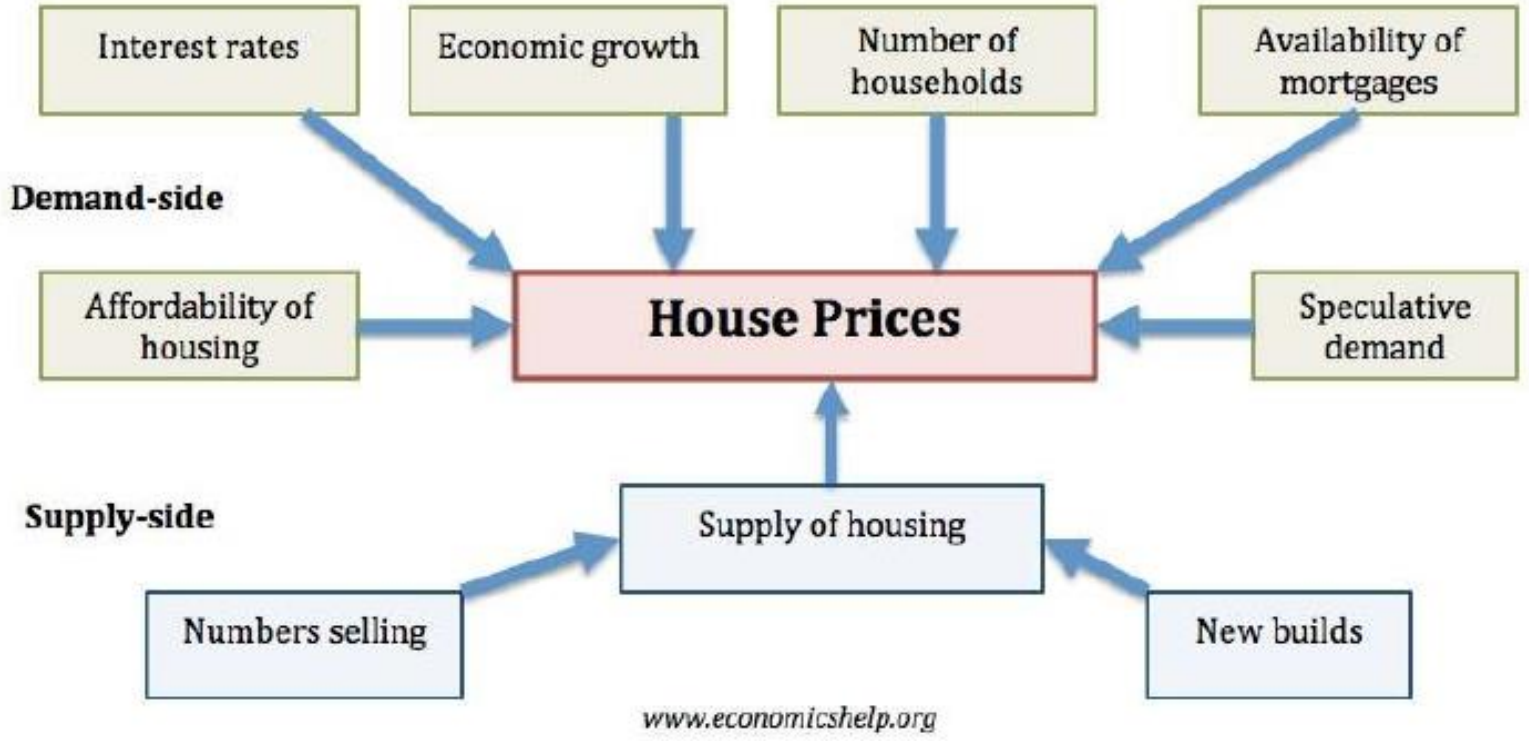


Figura 1 – Exemplo ilustrativo de variáveis temporais que podem definir "completamente" a dinâmica do preço de um imóvel residencial

Introdução

Matematicamente, o estado de um sistema dinâmico corresponde a um ponto no **espaço de estados**. Ou seja, cada coordenada está associada a uma variável de estado.

A **dinâmica** do sistema é a lei de variação do estado no tempo. Ela pode ser contínua ou discreta. A dinâmica contínua indica a direção de variação do estado, enquanto a dinâmica discreta indica o próximo estado.

Introdução

A **trajetória** descreve a evolução no tempo do estado do sistema. Em outras palavras, descreve o deslocamento de um ponto no espaço de estados regido pela dinâmica do sistema. A trajetória é parametrizada no tempo.

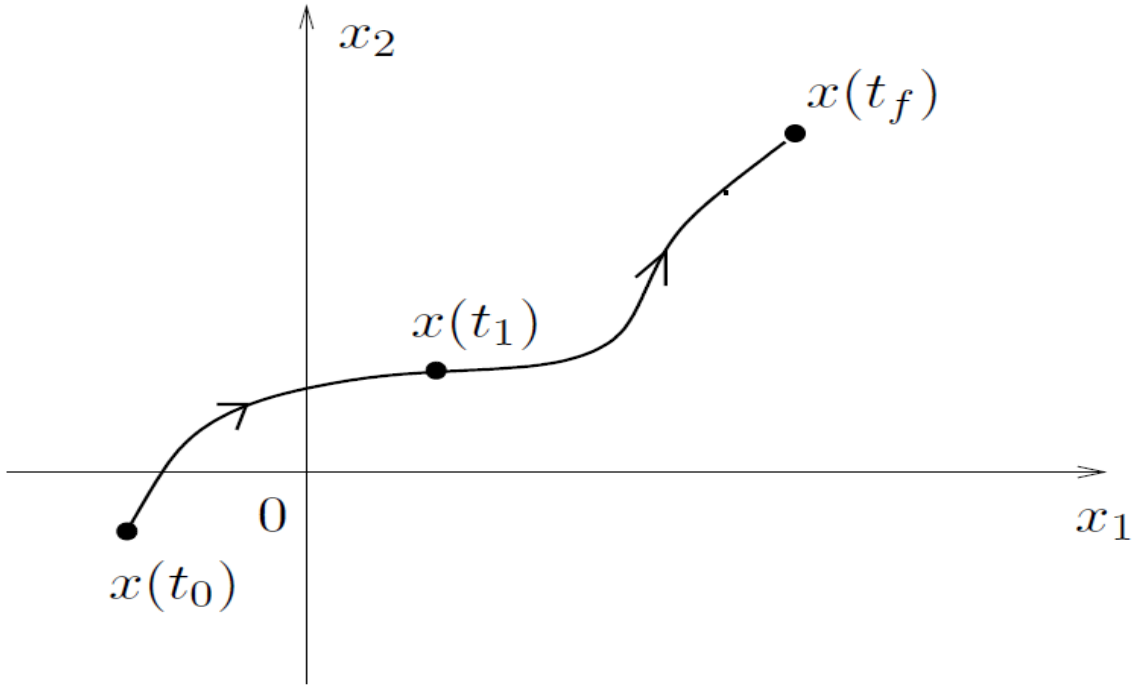


Figura 2 – Trajetória no espaço de estados para dinâmica contínua e estados contínuos

Introdução

A dinâmica pode ser estacionária (sistemas invariantes no tempo: quem não varia é a dinâmica, pois o estado pode variar no tempo) ou não-estacionária (sistemas variantes no tempo).

Cabe enfatizar o papel da entrada externa, quando presente:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_p \end{bmatrix} \quad \mathbf{f}(t, \mathbf{x}, \mathbf{u}) = \begin{bmatrix} f_1(t, \mathbf{x}, \mathbf{u}) \\ \vdots \\ f_n(t, \mathbf{x}, \mathbf{u}) \end{bmatrix} \quad \mathbf{g}(k, \mathbf{x}, \mathbf{u}) = \begin{bmatrix} g_1(k, \mathbf{x}, \mathbf{u}) \\ \vdots \\ g_n(k, \mathbf{x}, \mathbf{u}) \end{bmatrix}$$

Equações de estado:	$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \mathbf{u})$	$\mathbf{x}(k+1) = \mathbf{g}(k, \mathbf{x}, \mathbf{u})$
Equação de saída:	$\mathbf{y} = \mathbf{h}(t, \mathbf{x}, \mathbf{u})$	$\mathbf{y} = \mathbf{h}(t, \mathbf{x}, \mathbf{u})$

Supondo entrada externa constante ou ausente, a trajetória do sistema dinâmico se divide em transitório e regime permanente.

Diferente do que se enfatizou no caso da rede neural de Hopfield, por exemplo, onde havia ausência de entrada externa e foco no regime permanente, agora vamos nos concentrar em redes recorrentes com entrada externa e foco no transitório.

Backpropation ao longo do tempo (BPTT)

Considerando o caso de redes neurais recorrentes com realimentação de saída e presença de entrada externa, já vimos que é possível implementar uma técnica baseada em gradiente para ajuste dos pesos sinápticos.

A ideia é “desdobrar” a rede neural recorrente ao longo do tempo, produzindo arquiteturas como na Figura a seguir.

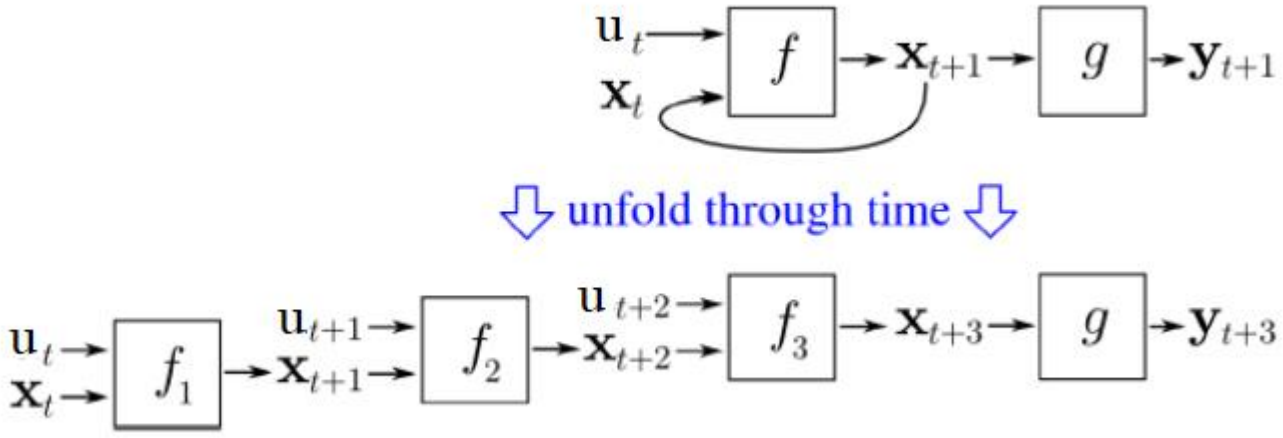


Figura 4 – Representação do desdobramento no tempo, para posterior aplicação do *backpropagation through time* (BPTT)

Backpropation ao longo do tempo (BPTT

Geralmente, uma etapa de BPTT é aplicada a cada *mini-batch*. No caso do bloco LSTM, *mini-batch* de tamanho 1 e 32 são comuns.

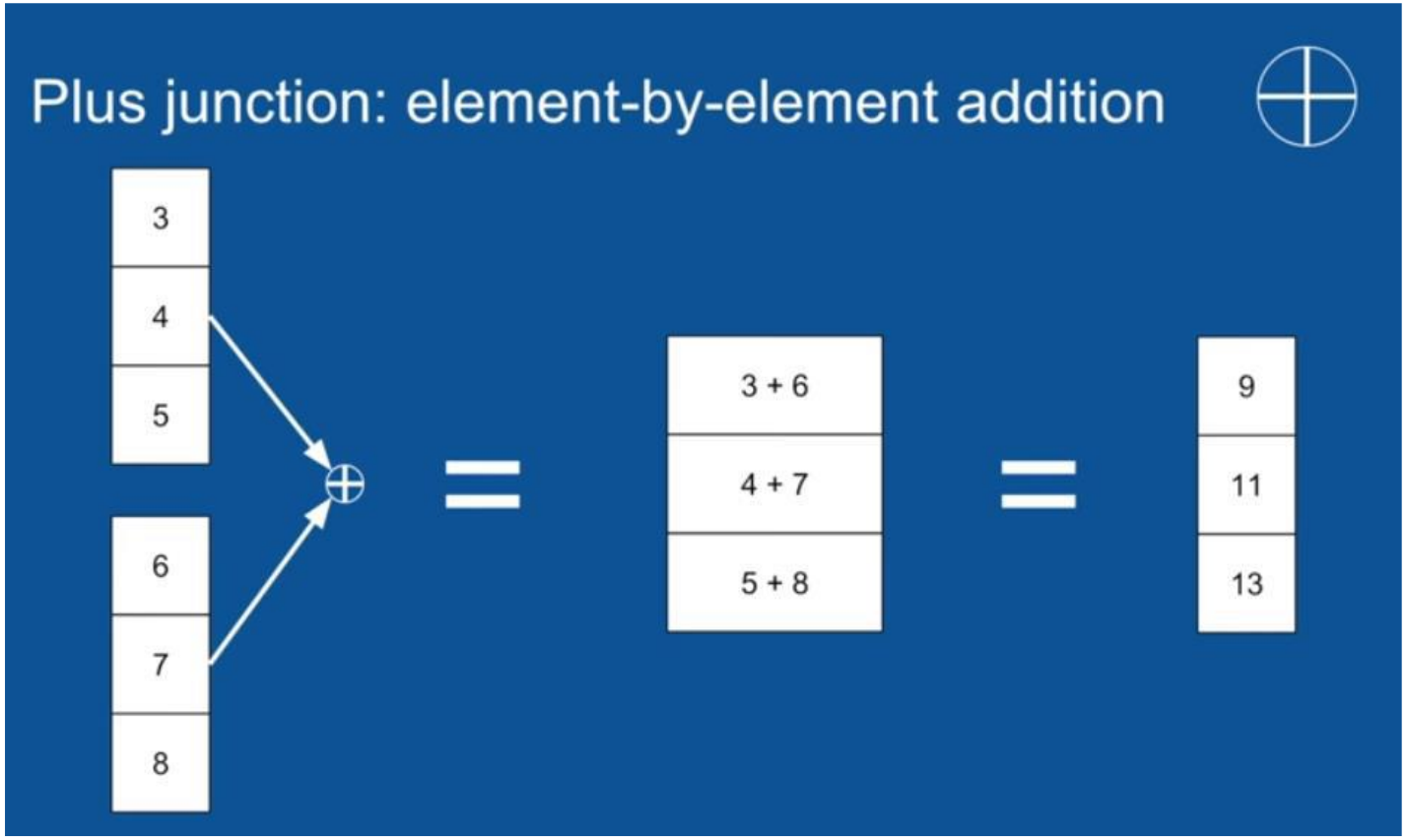
Os pesos associados ao mapeamento *f* são atualizados como a média dos pesos após a etapa de BPTT, tomando as *k* instâncias de *f*, sendo que no caso da Figura anterior tem-se $k = 3$.

Motivação para o bloco LSTM

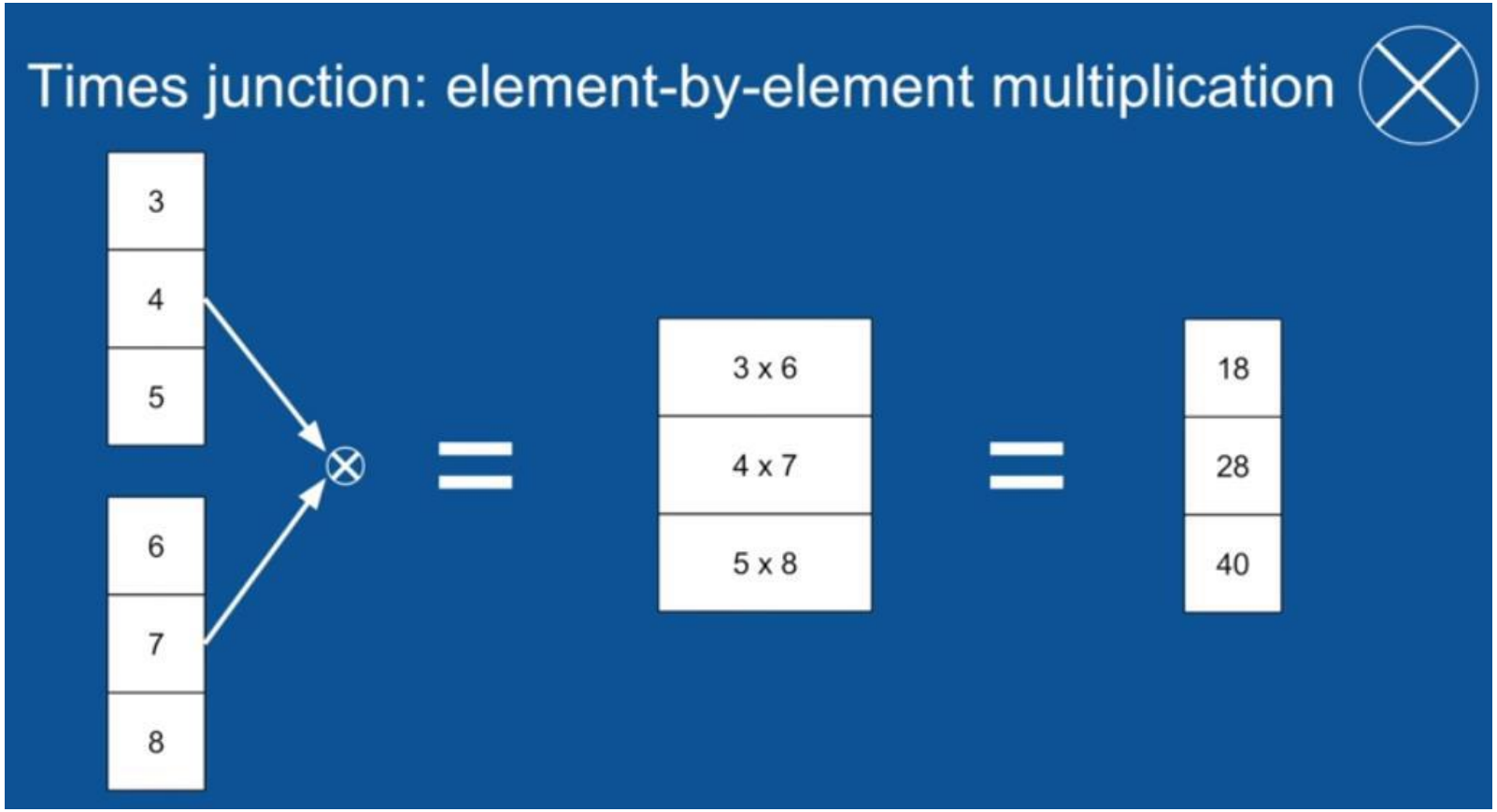
O bloco LSTM (*Long Short Term Memory*) foi proposto em 1997 (HOCHREITER & SCHMIDHUBER, 1997) e, como o próprio nome indica, foi concebido para viabilizar a implementação conjunta de memórias de longo e de curto prazos em redes neurais recorrentes. Essas memórias estão associadas aos estados internos.

Sugere-se a leitura do paper de revisão com as principais conquistas na área: GREFF et al. (2017).

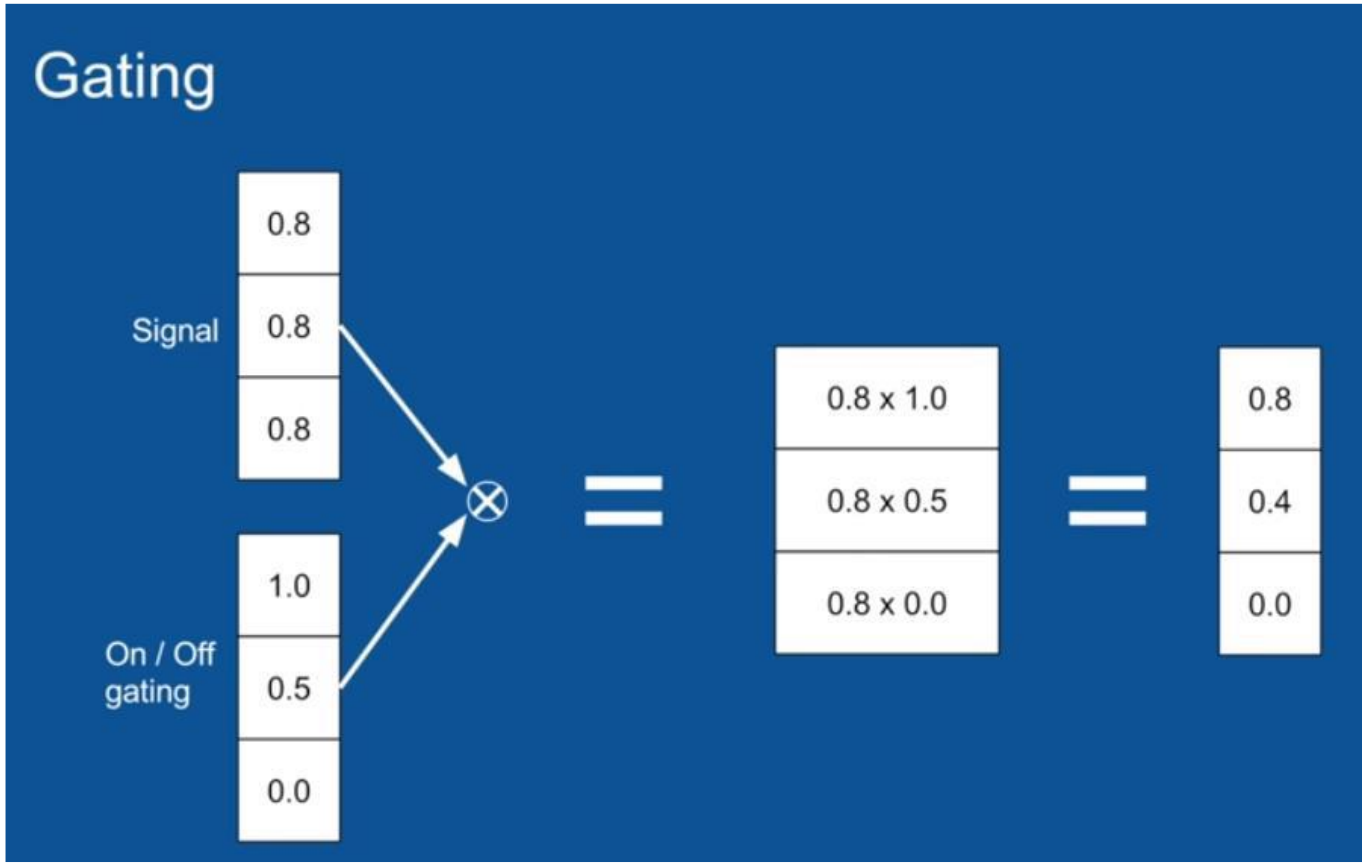
Compreendendo o bloco LSTM



Compreendendo o bloco LSTM



Compreendendo o bloco LSTM



Compreendendo o bloco LSTM

Redes neurais recorrentes apresentam realimentação de informação, o que significa que uma certa informação pode ou não persistir ao longo do tempo.

Para o caso específico de um bloco de neurônios que apresenta realimentação externa, é possível representar este bloco como uma sequência de múltiplas cópias de si mesmo, cada cópia fornecendo uma entrada ao bloco sucessor na sequência.

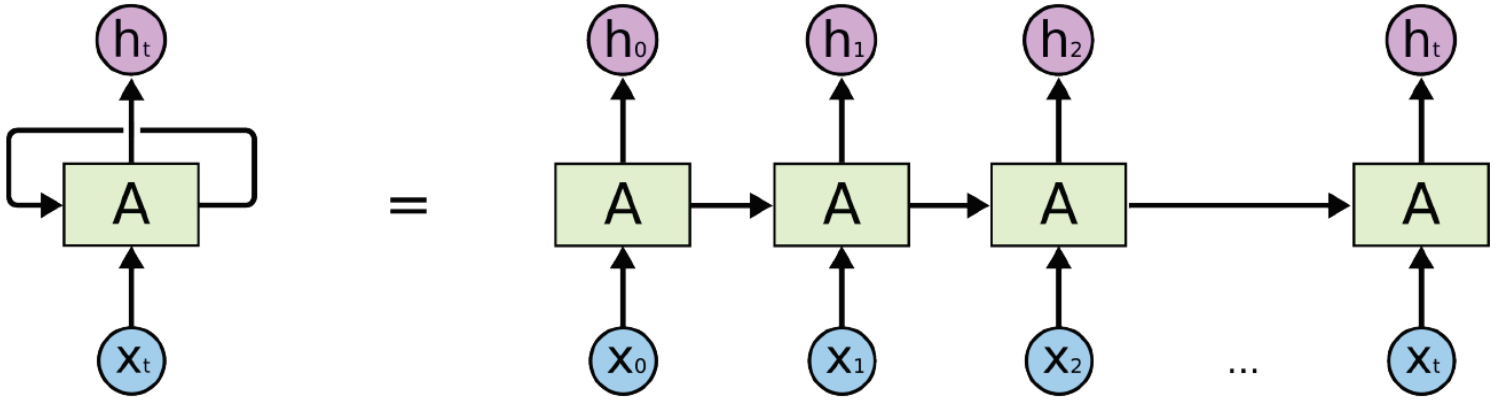


Figura 5 – Rede neural recorrente e o seu desdobramento no tempo

Compreendendo o bloco LSTM

Fica evidente, portanto, que redes neurais recorrentes estão intimamente vinculadas a aplicações que envolvem contexto e informação sequencial. De fato, há um elenco consistente e expressivo de aplicações bem-sucedidas de redes neurais recorrentes em reconhecimento de fala, modelagem de linguagem, tradução de textos e rotulação de imagens.

No entanto, até a proposição das LSTMs (*Long Short Term Memories*) em 1997 (HOCHREITER & SCHMIDHUBER, 1997), o ajuste de parâmetros em redes recorrentes convencionais se mostrava bem mais desafiador que no caso não-recorrente. Além disso, a capacidade de implementar simultaneamente memórias de curto, médio e longo prazos, necessária em várias aplicações, era bem limitada. Portanto, os blocos LSTMs são responsáveis por grande parte do sucesso recente das redes neurais recorrentes.

Redes formadas por um ou mais blocos LSTMs podem aprender dependências de prazos arbitrários, particularmente de longo prazo, sem muito esforço.

Compreendendo o bloco LSTM

Uma rede neural recorrente convencional apresenta um bloco na forma ilustrada na Figura 6, exibida na versão com desdobramento no tempo e tal que o módulo em amarelo realiza um mapeamento multidimensional não-linear.

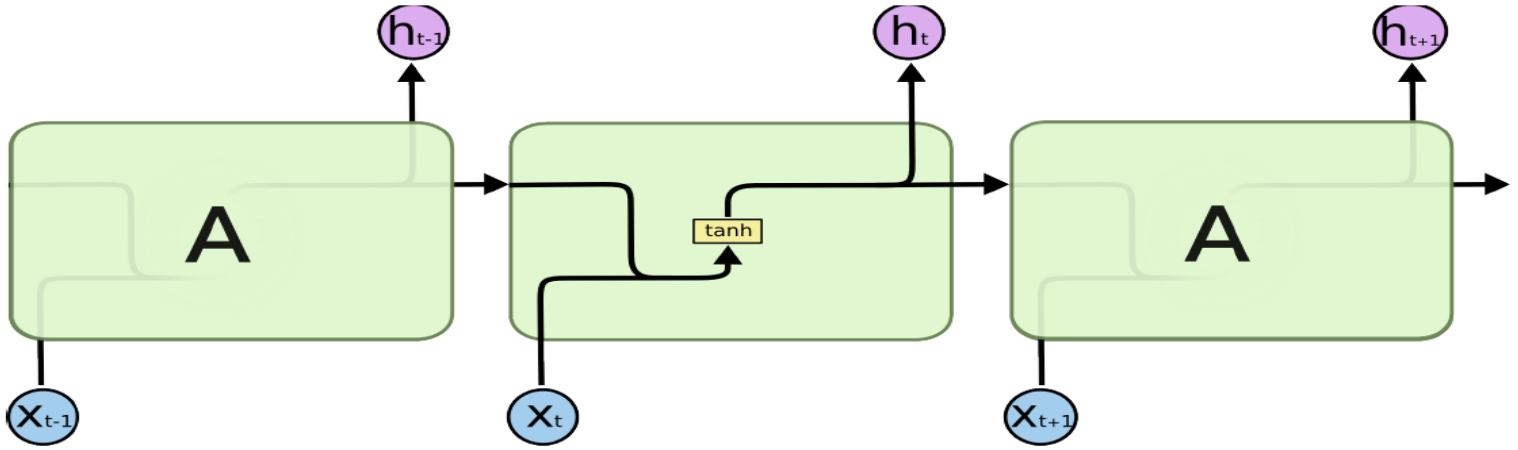


Figura 6 – Bloco de uma rede neural recorrente convencional com realimentação externa.

Repare que o bloco é bastante inflexível, pois só é capaz de preservar informações de curto prazo, mais especificamente, o sinal produzido no instante anterior.

Compreendendo o bloco LSTM

Portanto, para memórias de longo prazo, sua implementação ficava vinculada à existência de um número equivalente de blocos, em sequência (repare que não se está falando aqui do desdobramento no tempo e sim blocos recorrentes consecutivos). Logo, a duração da memória ficava associada à arquitetura. Mais ainda, supondo a necessidade de implementação conjunta de memórias de prazos diferentes, todos esses prazos deveriam ser contemplados a priori na arquitetura da rede neural, não sendo, portanto, passíveis de aprendizado de acordo com as demandas da aplicação.

Já o bloco LSTM (figura abaixo) possui uma estrutura mais sofisticada, composta por quatro mapeamentos multidimensionais não-lineares, sendo três deles portas (*gates*) que permitem implementar filtros e memórias de prazo arbitrário (**e com um prazo específico para cada dimensão do vetor de sinais**) em um único bloco LSTM.

Compreendendo o bloco LSTM

De acordo com a notação apresentada, o **fluxo de sinais é vetorial** e as operações são realizadas **elemento a elemento** do vetor de sinais. Além disso, os retângulos amarelos são mapeamentos multidimensionais não-lineares.

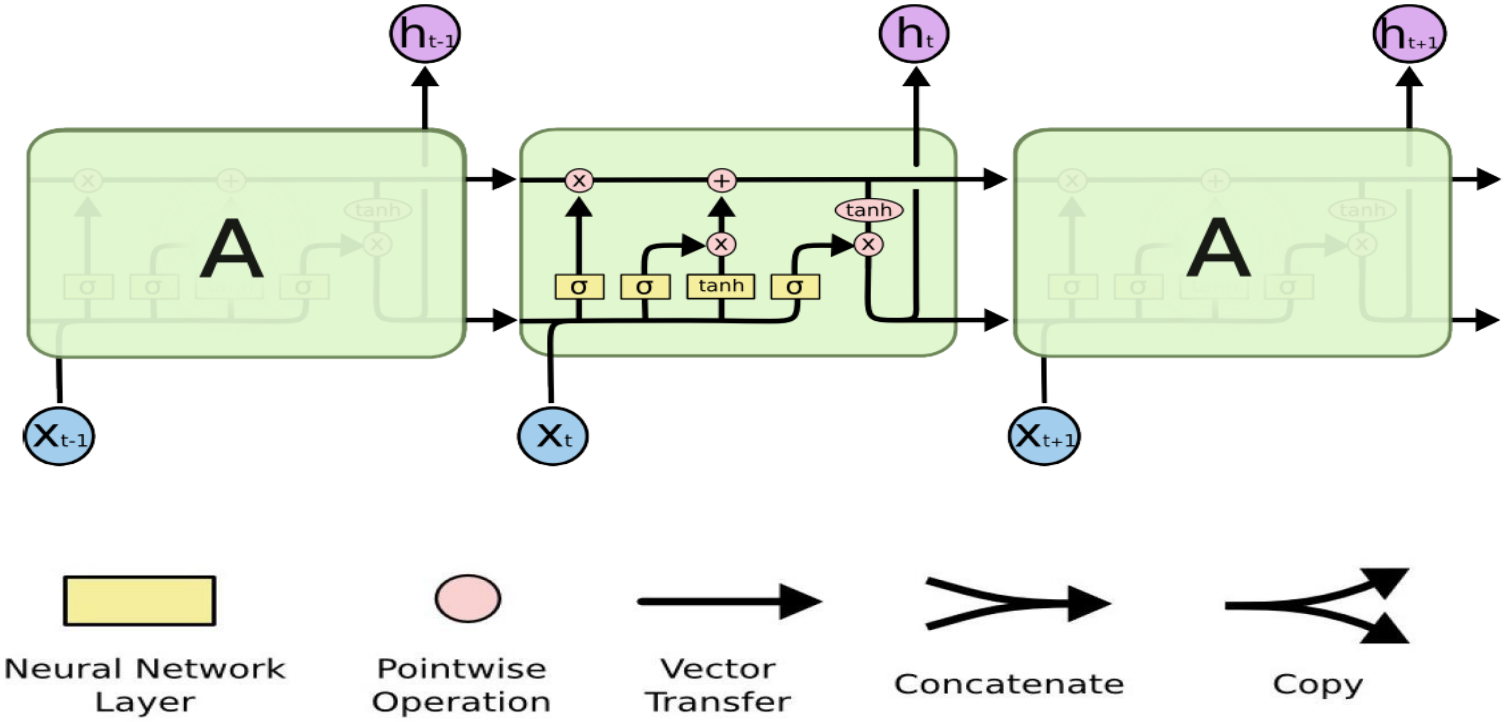


Figura 7 – Bloco LSTM e sua notação associada

Compreendendo o bloco LSTM

Como em toda estrutura recorrente, o aspecto-chave do bloco LSTM é o seu vetor de estados, que corresponde ao sinal presente na linha horizontal em destaque na figura abaixo

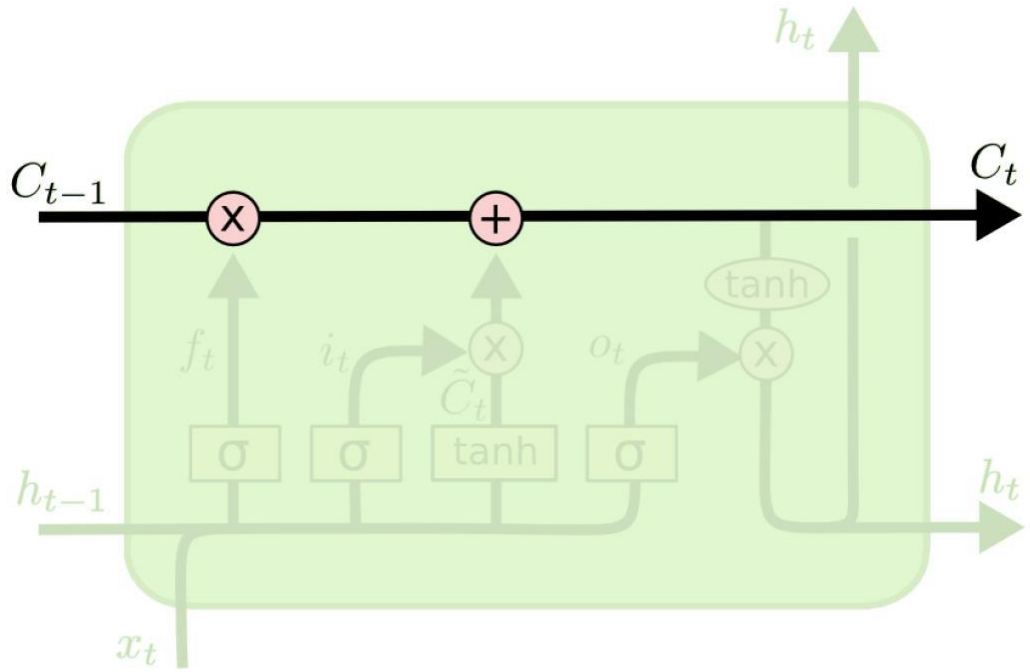
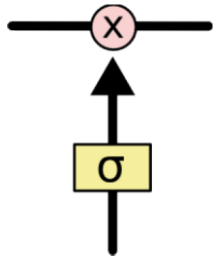


Figura 8 – Destaque para o vetor de estados do bloco LSTM

Compreendendo o bloco LSTM

O vetor de estados pode passar por três operações algébricas simples, permitindo remover informação do vetor de estados ou adicionar informação ao vetor de estados. Logo, caso nenhuma remoção ou adição ocorra, o vetor de estados do bloco LSTM se mantém inalterado de um instante de tempo ao seguinte.

Essas operações sobre o vetor de estados do bloco LSTM são comandadas por três portas cujo comportamento é definido durante o processo de treinamento.

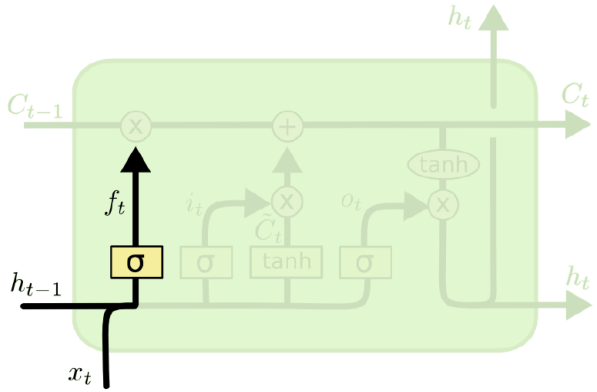


Cada elemento do vetor de saída de uma porta pode assumir um valor no intervalo [0,1]. Valores iguais a 0 removem toda a informação daquele canal, enquanto que valores iguais a 1 mantêm toda a informação naquele canal.

O passo-a-passo do bloco LSTM

O papel da primeira porta é indicar o que deve ser preservado para o instante *t* do vetor de estados C_{t-1} do bloco LSTM, elemento a elemento.

Dadas as entradas *ht*−1 e *xt*, elas são mapeadas no vetor de saídas *ft*, onde cada um de seus elementos corresponde a um valor no intervalo [0,1]. Esta porta é denominada de porta do esquecimento (*forgetting gate*).

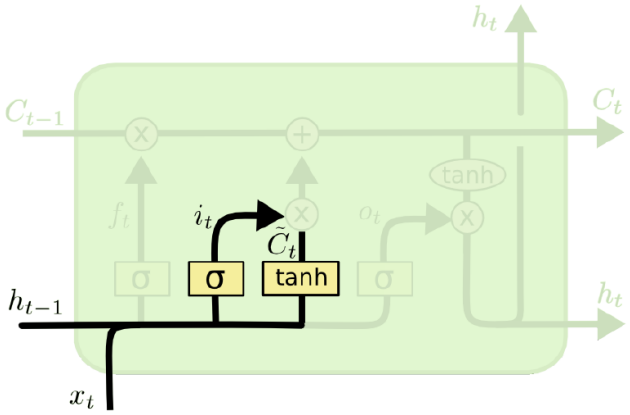


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figura 9 – Destaque para a porta do esquecimento

O passo-a-passo do bloco LSTM

O próximo passo é decidir que informação nova será adicionada ao vetor de estados C_{t-1} para se produzir C_t. Isso é feito em duas partes: (1) Uma porta decide em que grau (valor no intervalo [0,1]) os elementos do vetor de estados serão atualizados, sendo denominada porta de entrada (foi chamada de *ignoring gate* na seção anterior); (2) Uma camada de neurônios com função de ativação tangente hiperbólica fornece a informação nova $\tilde{C}_t$ (valor no intervalo [−1,+1]) a ser adicionada ao vetor de estados, de acordo com a indicação da porta de entrada.



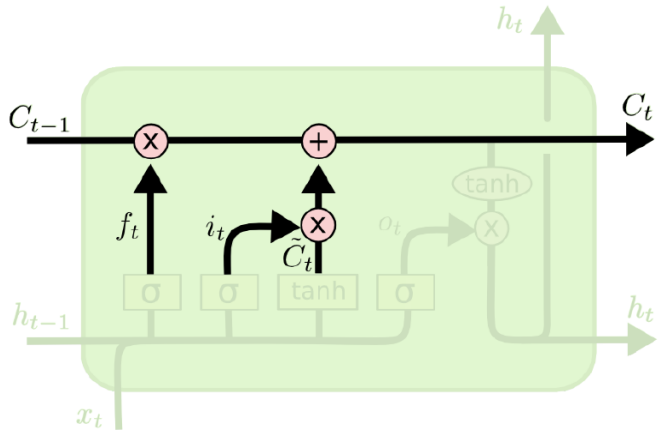
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figura 10 – Destaque para a porta de entrada e para a nova informação

O passo-a-passo do bloco LSTM

O novo vetor de estados *Ct* do bloco LSTM é definido pela **composição aditiva** das operações das Figuras 9 e 10, conforme indicado na Figura 11.

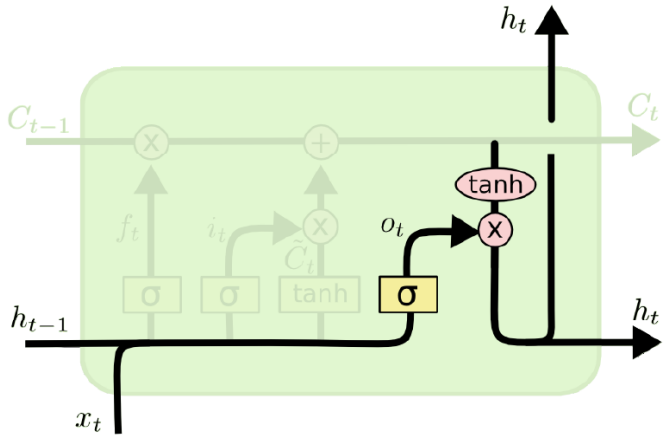


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figura 11 – Definição do novo vetor de estados

A saída do bloco LSTM não vai ser diretamente o seu novo vetor de estados. Primeiro, o vetor de estados C_t é mapeado para o intervalo [−1,+1]. Em seguida, a porta de saída (foi chamada de *selection gate* na seção anterior) vai indicar com que grau (valor no intervalo [0,1]) cada elementos de C_t será liberado

O passo-a-passo do bloco LSTM



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

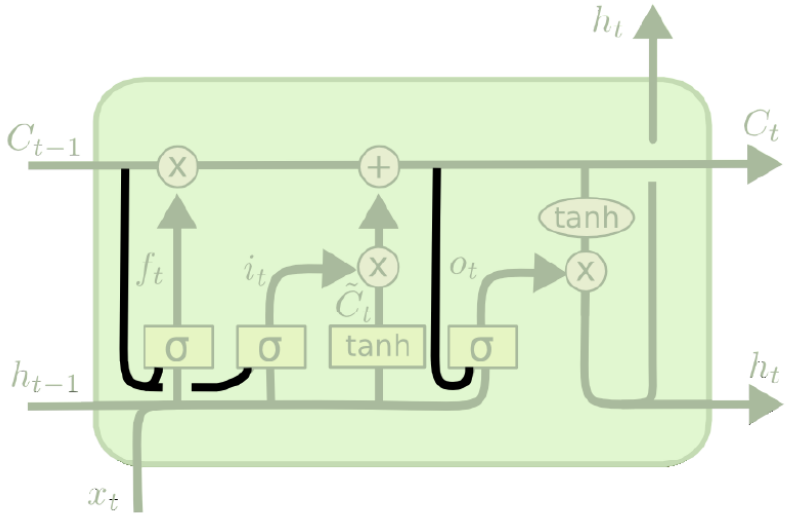
Figura 12 – Destaque para a porta de saída e para o sinal de saída h_t efetivamente liberado pelo bloco LSTM

Variantes do bloco LSTM

Existem propostas na literatura que diferem da versão convencional apresentada até aqui, ao inserir fluxos adicionais de informação. O propósito é trazer ganhos em aplicações específicas, mas à custa de um maior desafio computacional na fase de treinamento.

Variantes do bloco LSTM

Uma variação bastante utilizada é aquela que acopla os graus de esquecimento e de inovação, ou seja, acopla os papéis das portas de esquecimento e de entrada, de modo que elas operem de modo complementar. Com isso, cada elemento do vetor de estados vai estar sujeito a uma operação complementar entre esquecimento e inovação.

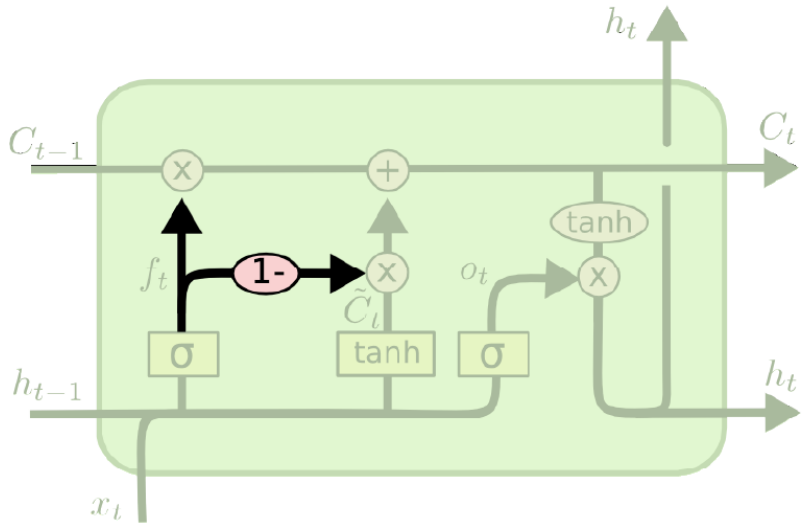


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Figura 13 – Bloco LSTM que considera o vetor de estados como uma entrada adicional para as três portas

Variantes do bloco LSTM

Neste caso, só há esquecimento caso haja informação nova a ser inserida em seu lugar. Em outras palavras, só se adiciona conteúdo novo caso haja conteúdo antigo a ser esquecido, na mesma proporção.



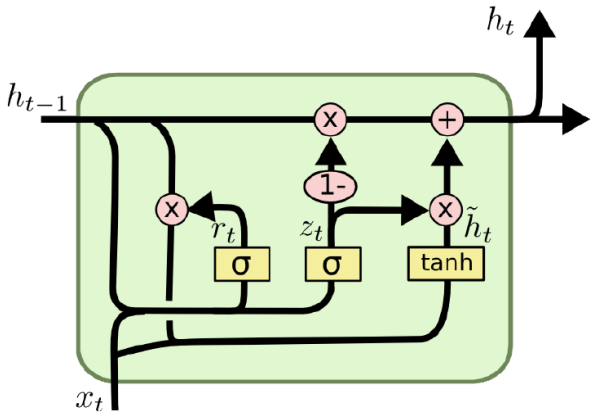
$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

Figura 14 – Bloco LSTM com operação complementar entre as portas de esquecimento e de entrada

Variantes do bloco LSTM

Uma variação mais acentuada do bloco LSTM é denominada GRU (*Gated Recurrent Unit*) e foi introduzida em 2014 (CHO et al., 2014). Seu treinamento é mais simples e esta proposta vem se tornando bastante popular, devido aos bons resultados comparativos.

A GRU combina as portas de esquecimento e de entrada numa porta de atualização e também faz com que o vetor de estados passe a ser o vetor de saídas. Outras modificações também estão presente



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figura 15 – Bloco GRU (*Gated Recurrent Unit*)

Aspectos de implementação

Já foi evidenciado que o treinamento de redes neurais recorrentes é desafiador, embora a tarefa tenha sido bem facilitada no caso do bloco LTSM, inclusive com a atenuação do efeito dos gradientes explosivos e que se anulam. Mesmo assim, particularmente com o aumento no número de pesos sinápticos e levando em conta o desdobramento no tempo (*backpropagation through time*), se verificam, na prática, limitações impostas pela arquitetura padrão von Neumann.

Iniciativas têm sido apresentadas na literatura para contornar essas limitações: Li, C., Wang, Z., Rao, M. et al. “Long short-term memory networks in memristor crossbar arrays”, Nature Machine Learning, vol. 1, pp. 49-57, 2019 (https://doi.org/10.1038/s42256-018-0001-4).

Um livro-texto relevante voltado para códigos em Python (API: Keras): Brownlee, J. “Long Short-Term Memory Networks With Python”, 2017.

Referências bibliográficas

CHO, K., VAN MERRIENBOER, B., GULCEHRE, C., BAHDANAU, D., BOUGARES, F., SCHWENK, H., BENGIO, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, arXiv:1406.1078, 2014.

GREFF, K., SRIVASTAVA, R.K., KOUTNÍK, J., STEUNEBRINK, B.R., SCHMIDHUBER, J. LSTM: A Search Space Odyssey, *IEEE Trans. on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222-2232, 2017.

HOCHREITER, S., SCHMIDHUBER, J. Long short-term memory, Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997