REDE NEURAL RECORRENTE

Prof. Valmir Macário Filho

DC - UFRPE

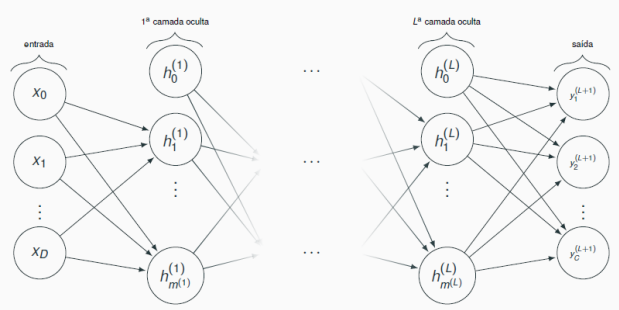
VISÃO GERAL

- Conceitos Básicos
- Redes de Elman
- Aplicações
- Treinamento das RNNS
- Dissipação/explosão dos gradientes

CONCEITOS BÁSICOS

REDES DE PROPAGAÇÃO FEEDFOWARD

- Redes de propagação adiante (feedforward networks) consideram que
  - os exemplos de treinamento são IID,
  - entrada e saída têm tamanho fixo.



De que forma considerar dados de tamanhos variados em que há dependências de curto/longo prazo?

MOTIVAÇÃO

- Nem todas as entradas podem ser convertidas para um vetor de tamanho fixo
- Problemas como reconhecimento de fala ou previsão de séries temporais exigem que um sistema armazene e use informações de contexto
  - Exemplo simples: A saída é SIM se número de 1s é par, senão é NÃO
  - 1000010101 – YES, 100011 – NO, …
- Difícil/Impossível escolher uma janela de contexto fixa
  - Sempre pode haver uma nova amostra mais do que qualquer coisa vista

REDES RECORRENTES

- Constituem uma ampla classe de redes cuja evolução do estado depende tanto da entrada corrente quanto do estado atual.
- São sistemas Turing-completos (*Turing complete*).
  - i.e., com tempo, dados e neurônios suficientes, RNNs podem aprender qualquer coisa que um computador pode fazer.
- Aplicáveis a dados sequenciais
  - Texto, vídeo, áudio, séries temporais ...

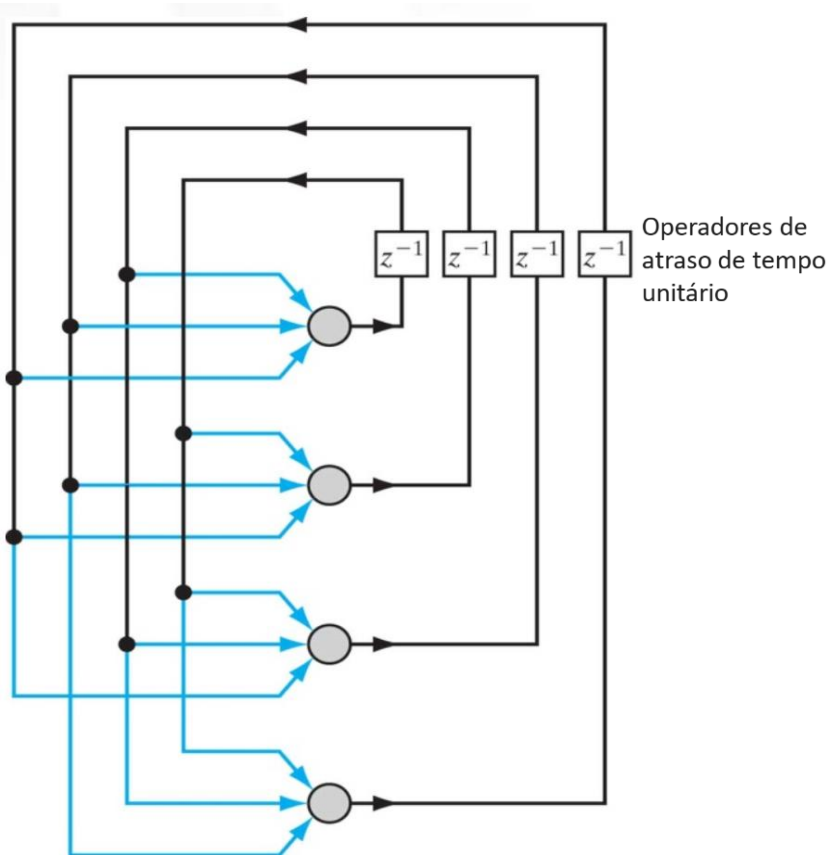
ALGUNS MODELOS HISTÓRICOS

- Rede de Hopfield: proposta por John Hopfield em 1982
- Rede de Jordan: proposta por Michael I. Jordan em 1986
- Rede de Elman: proposta por Jeffrey L. Elman em 1990

REDE DE HOPFIELD

$$w_{ij} = w_{ji}$$
$$w_{ii} = 0$$

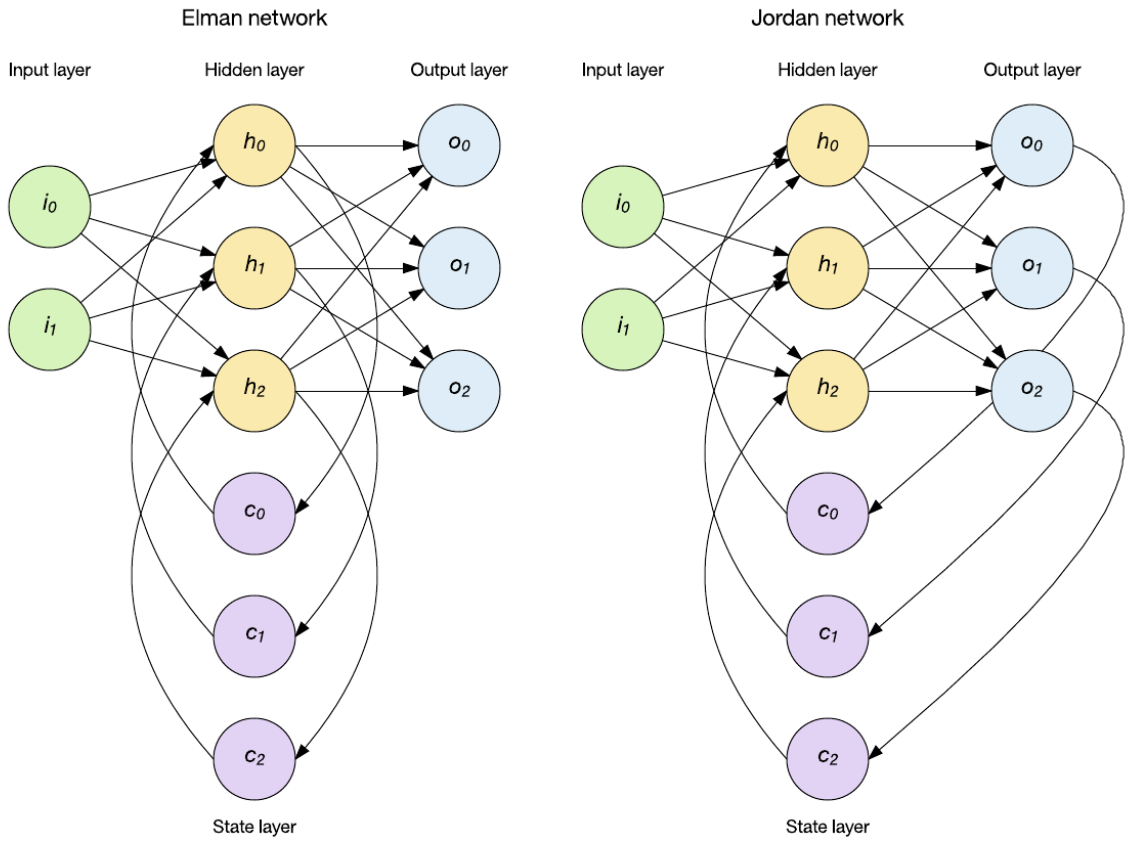
- Única camada de neurônios totalmente e simetricamente interconectados a partir de operadores de atraso unitário, não havendo conexões de um neurônio para ele mesmo
- Memória autoassociativa sem camada escondida
- A aprendizagem realizada em um único passo
- Na fase de uso, a propagação de uma entrada requer múltiplos passos até que um estado estável seja atingido (saída da rede)



REDES DE ELMAN E JORDAN

- As redes Jordan mantém o histórico da camada de saída na memória;
- As redes de Elman mantém o histórico da saída das camadas ocultas na memória;
- Podem ser treinadas com o algoritmo de backpropagation

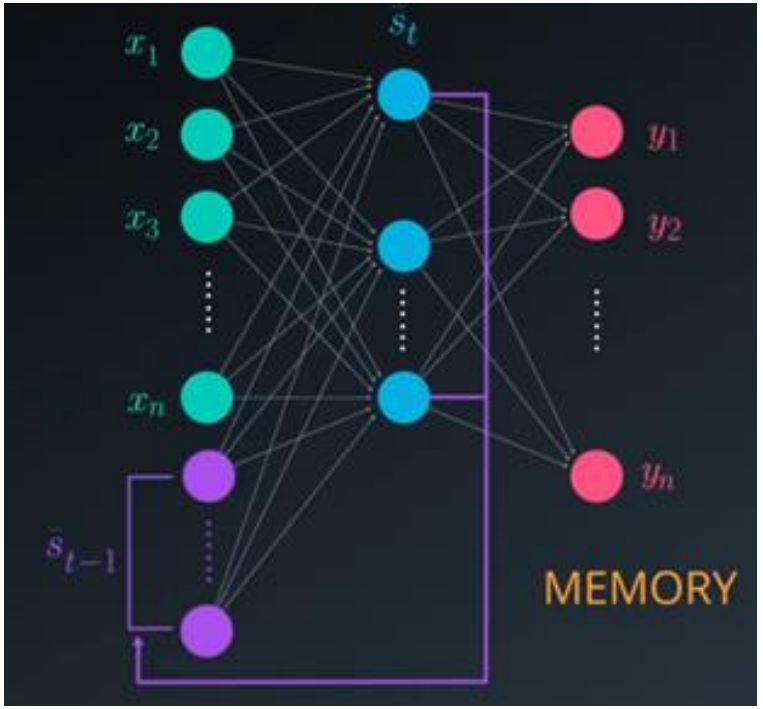
REDES DE ELMAN E JORDAN



REDES DE ELMAN

- Existem duas diferenças principais entre FFNNs e RNNs. A rede neural recorrente usa:
  - **sequências** como entradas na fase de treinamento, e
  - elementos de **memória**

- Memória é definida como a saída de neurônios de camadas ocultas, que servirão como entradas adicionais para a rede durante os próximos passos de treinamento.



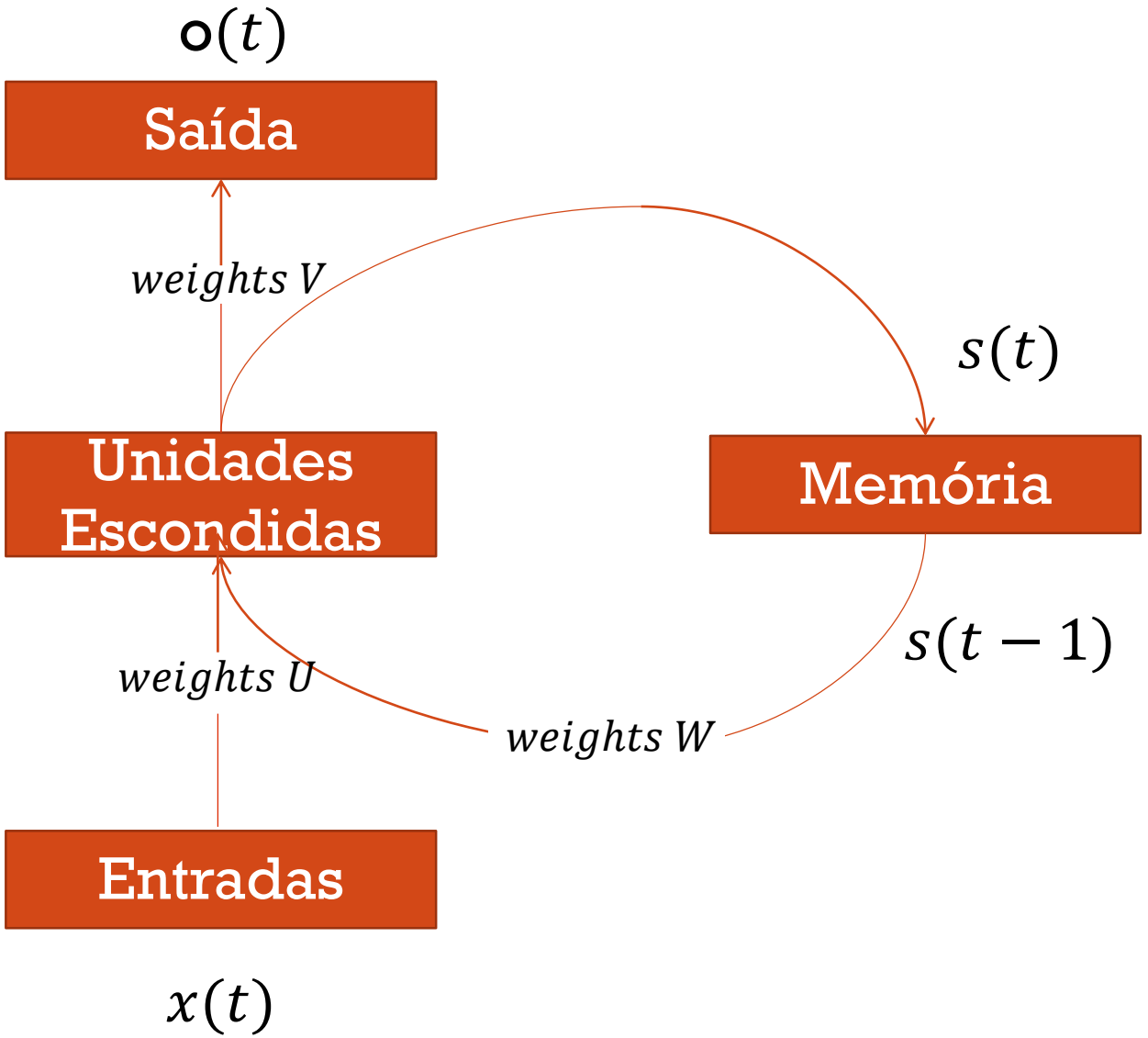
REDES NEURAIS RECORRENTES (RNNS)

- Redes Neurais Recorrentes tem a saída anterior ou estados ocultos como entradas.

- A entrada composta no momento t tem alguma informação histórica sobre os acontecimentos no momento T < t

- As RNNs são úteis porque seus valores intermediários (estado) podem armazenar informações sobre entradas passadas por um tempo que não é fixado a priori

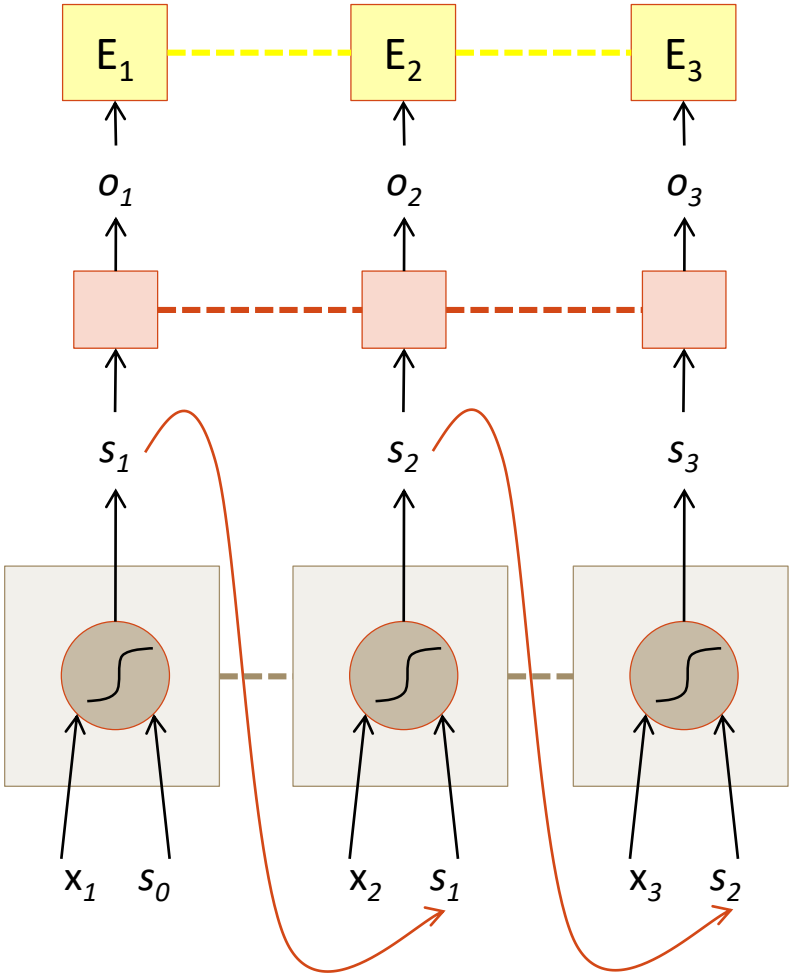
ARQUITETURA RNN



ARQUITETURA RNN

- O estado oculto s_t representa a memória da rede.
- s_t captura informações sobre o que aconteceu em todas as etapas de tempo anteriores.
- A saída ot é calculada exclusivamente com base na memória no tempo t.
- Normalmente, não é possível capturar informações de muitas etapas de tempo atrás.
- Ao contrário de uma rede neural profunda tradicional, que usa parâmetros diferentes em cada camada, uma RNN compartilha os mesmos parâmetros (U, V, W) em todas as etapas.
- Isso reflete o fato de que estamos executando a mesma tarefa em cada etapa, apenas com entradas diferentes. Isso reduz muito o número total de parâmetros que a rede precisa aprender.

RNN FORWARD



------ Pesos compartilhados

RECURRENT NEURAL NETWORKS (RNNS)

- Observe que os pesos são compartilhados ao longo do tempo

- Essencialmente, as cópias da célula RNN são feitas ao longo do tempo (desenrolamento/desdobramento), com diferentes entradas em diferentes etapas de tempo

EXEMPLOS DE APLICAÇÕES

CLASSIFICAÇÃO DE SENTIMENTOS

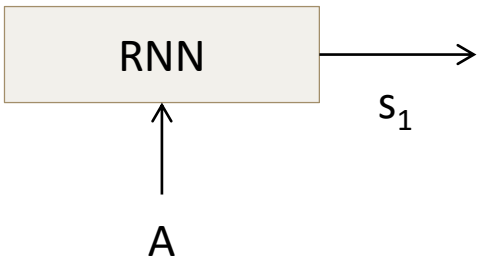
- Classifique uma
  - crítica de restaurantes no TripAdivisor OU
  - crítica de filmes do IMDB

…
como positiva ou negativa

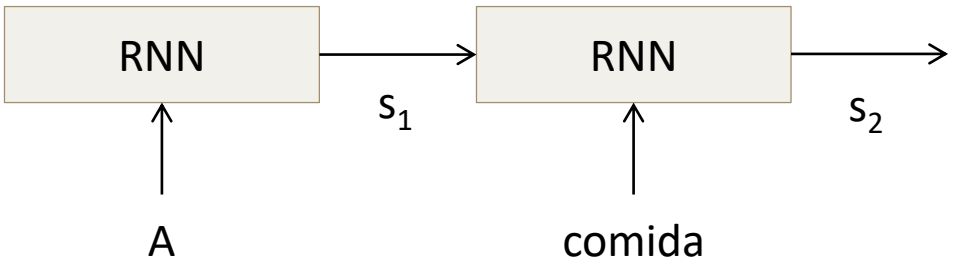
- Entradas: Muitas palavras, uma ou várias sentenças
- Saídas: Positiva/Negativa

- “A comida estava realmente muito boa”
- “O frango atravessou a estrada porque não estava cozido”

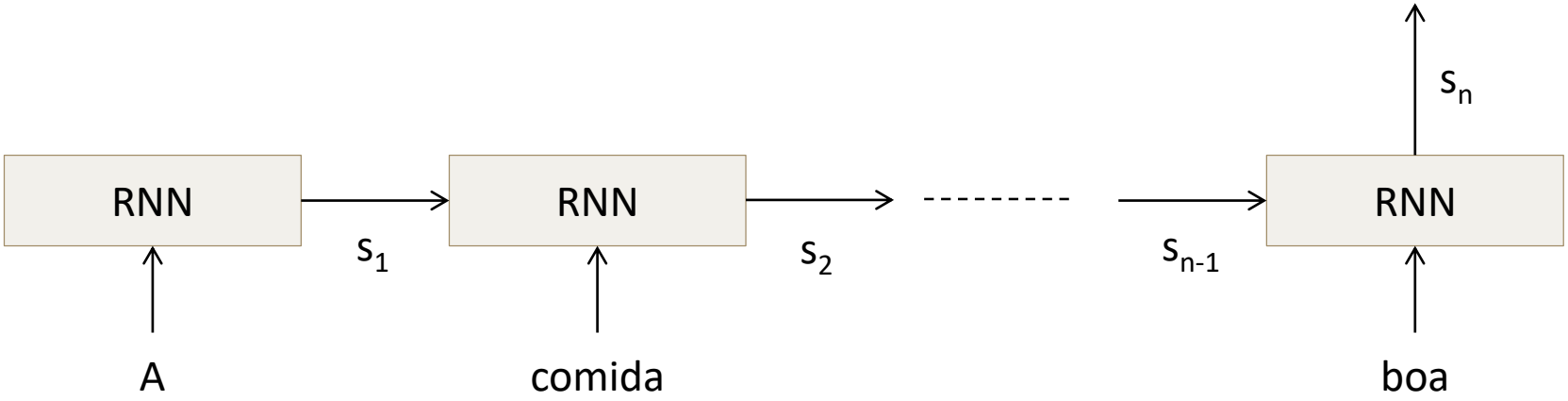
CLASSIFICAÇÃO DE SENTIMENTOS



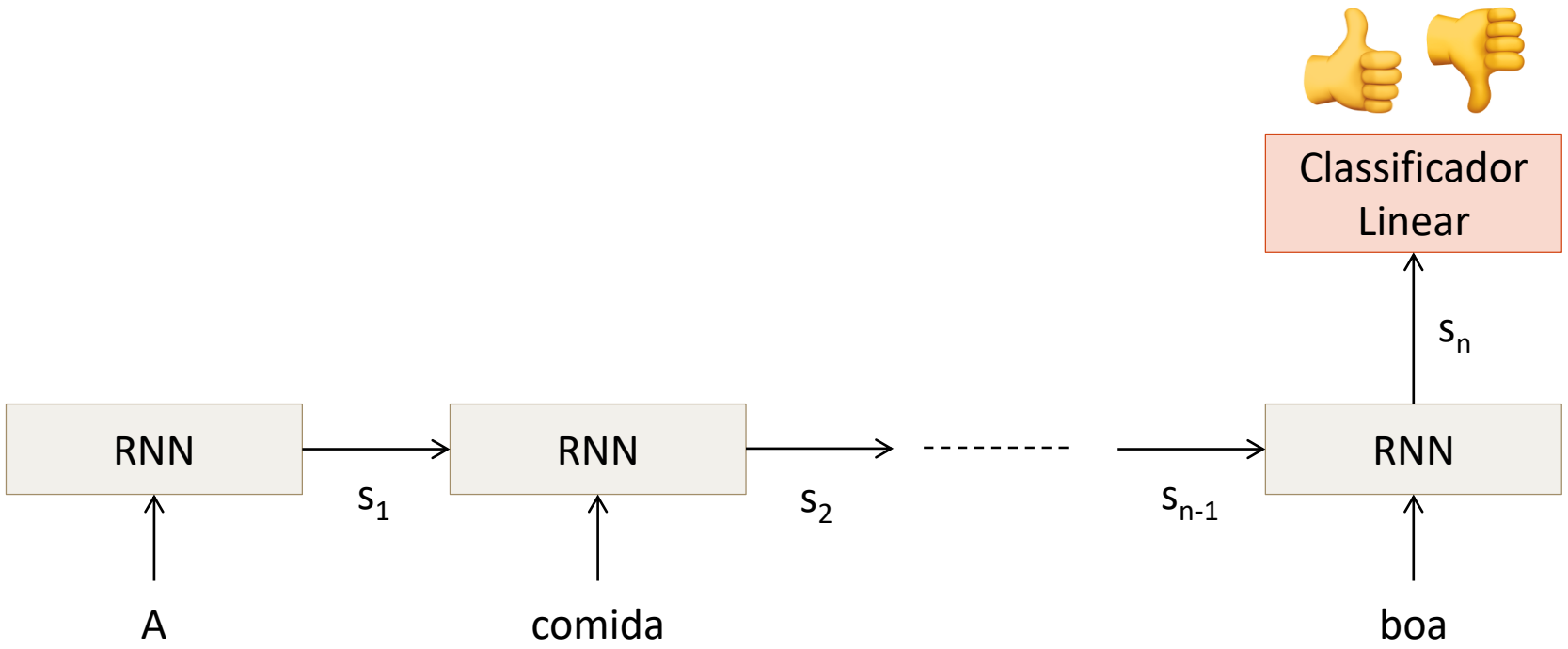
CLASSIFICAÇÃO DE SENTIMENTOS



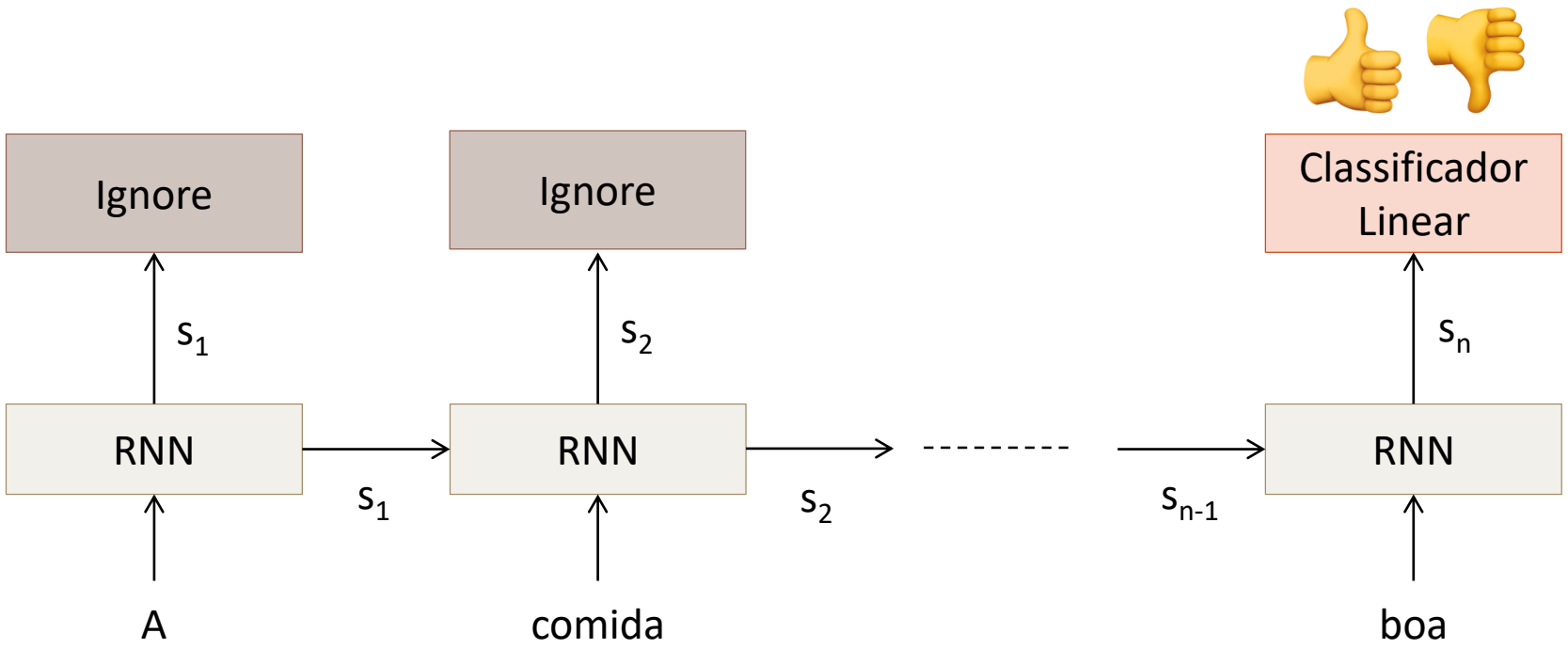
CLASSIFICAÇÃO DE SENTIMENTOS



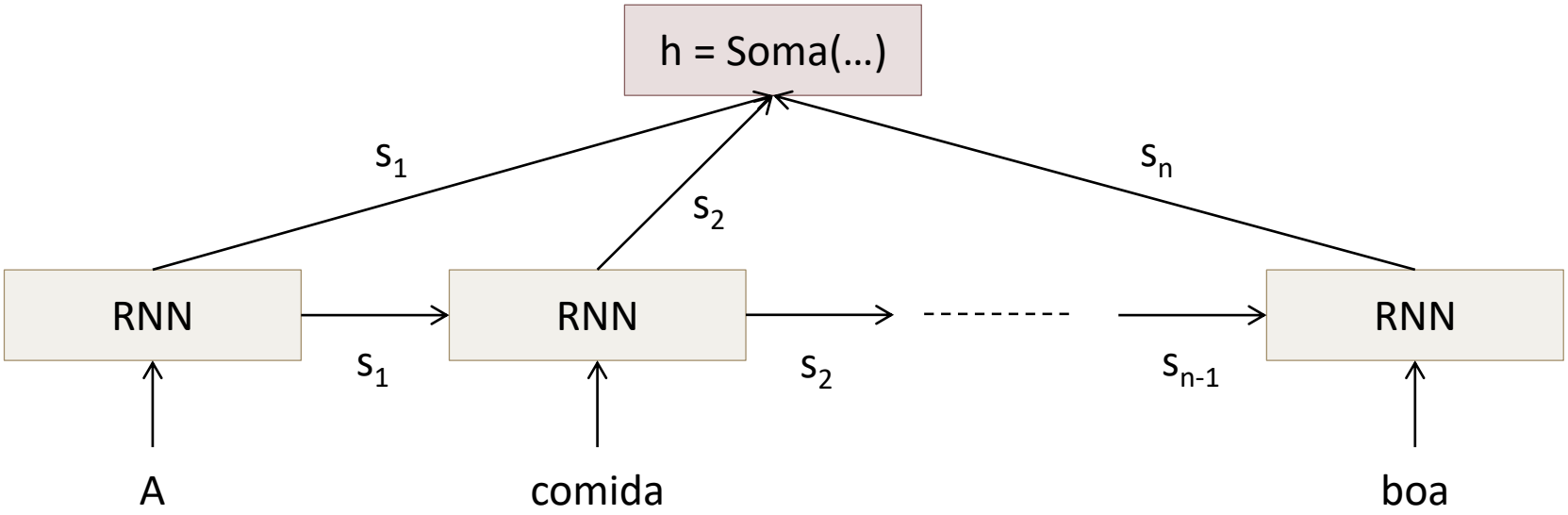
CLASSIFICAÇÃO DE SENTIMENTOS



CLASSIFICAÇÃO DE SENTIMENTOS

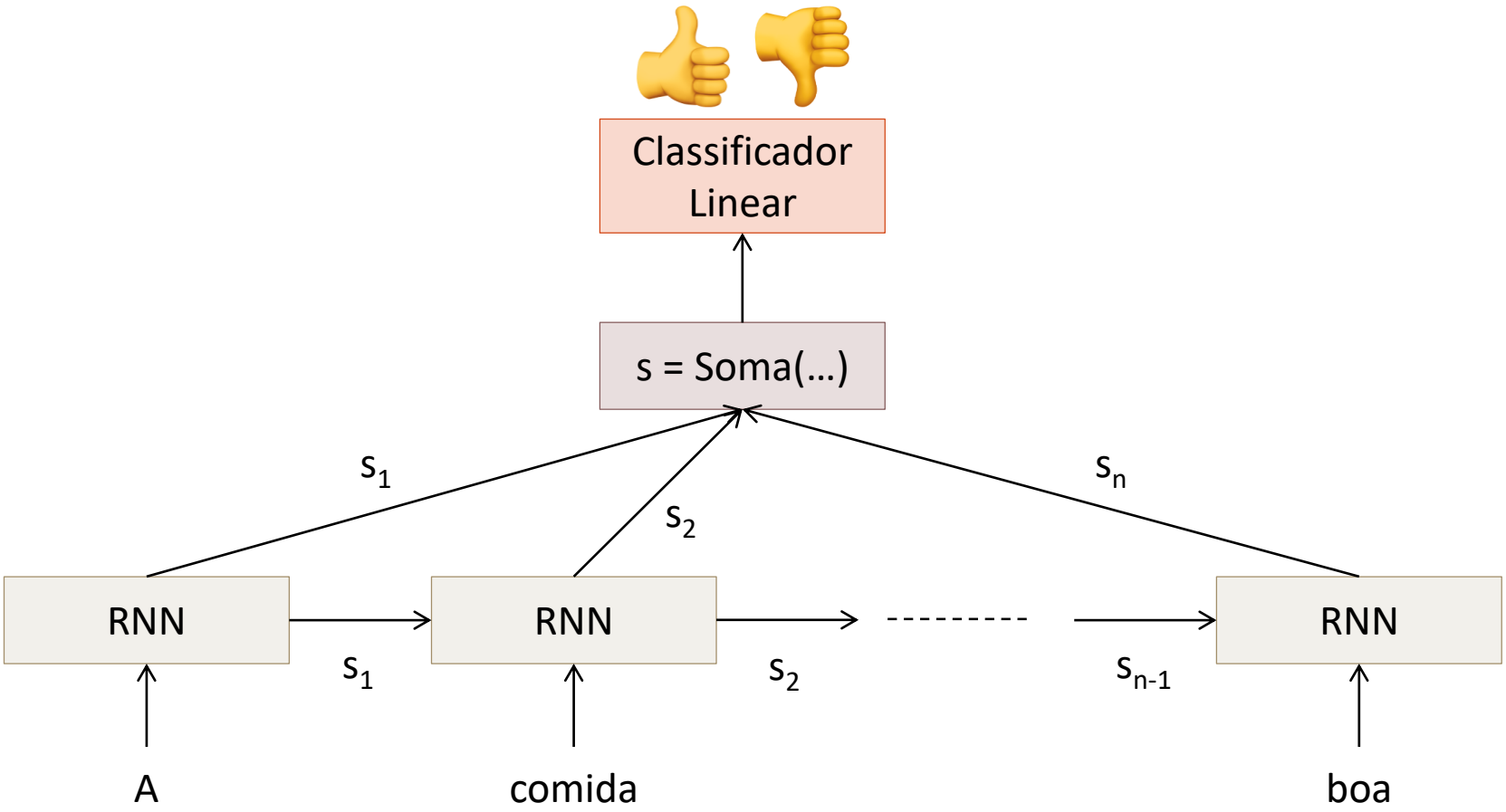


CLASSIFICAÇÃO DE SENTIMENTOS



http://deeplearning.net/tutorial/lstm.html

CLASSIFICAÇÃO DE SENTIMENTOS



http://deeplearning.net/tutorial/lstm.html

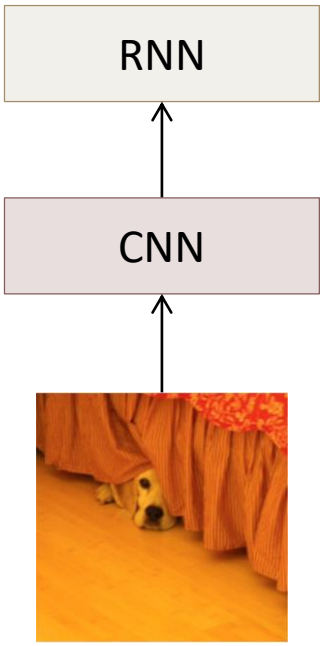
LEGENDAGEM DE IMAGEM (IMAGE CAPTIONING)

- Dada uma imagem, produza uma frase descrevendo seu conteúdo

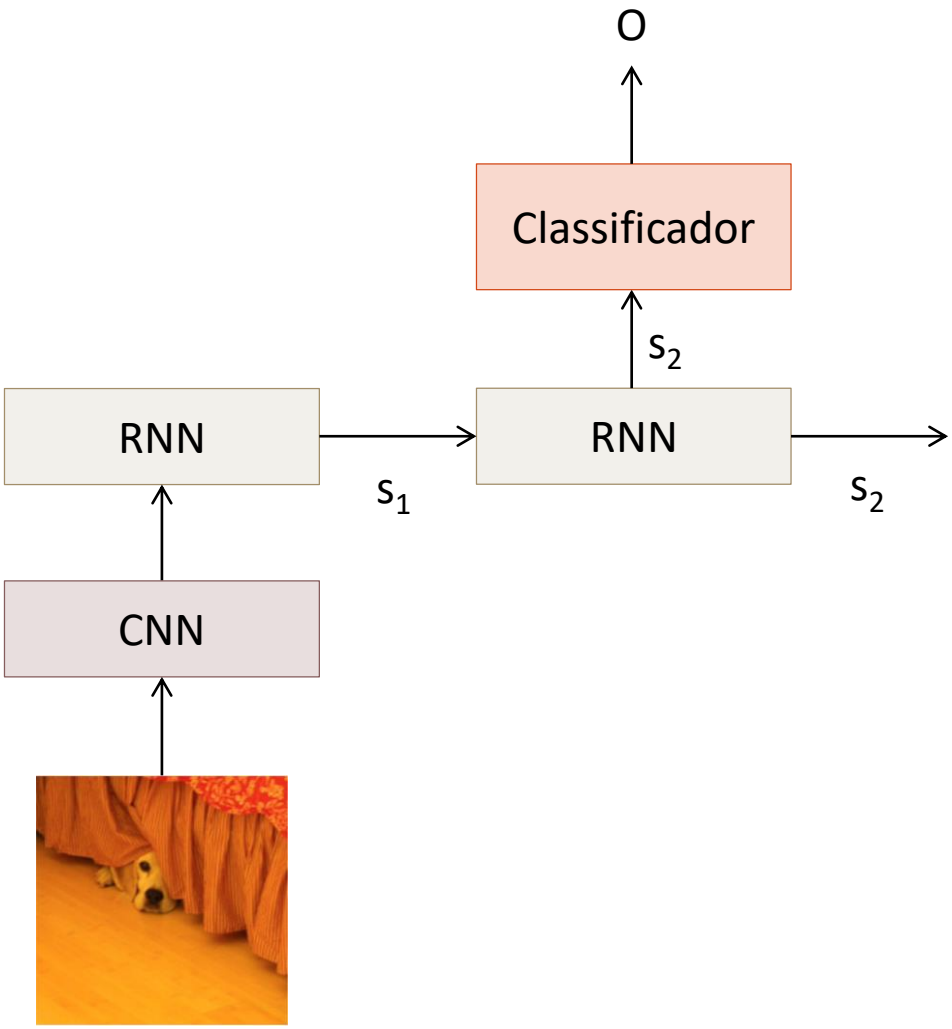
- Inputs: Características de uma imagem (De uma CNN)
- Outputs: Palavras (considere uma sentença)

: O cão está escondido

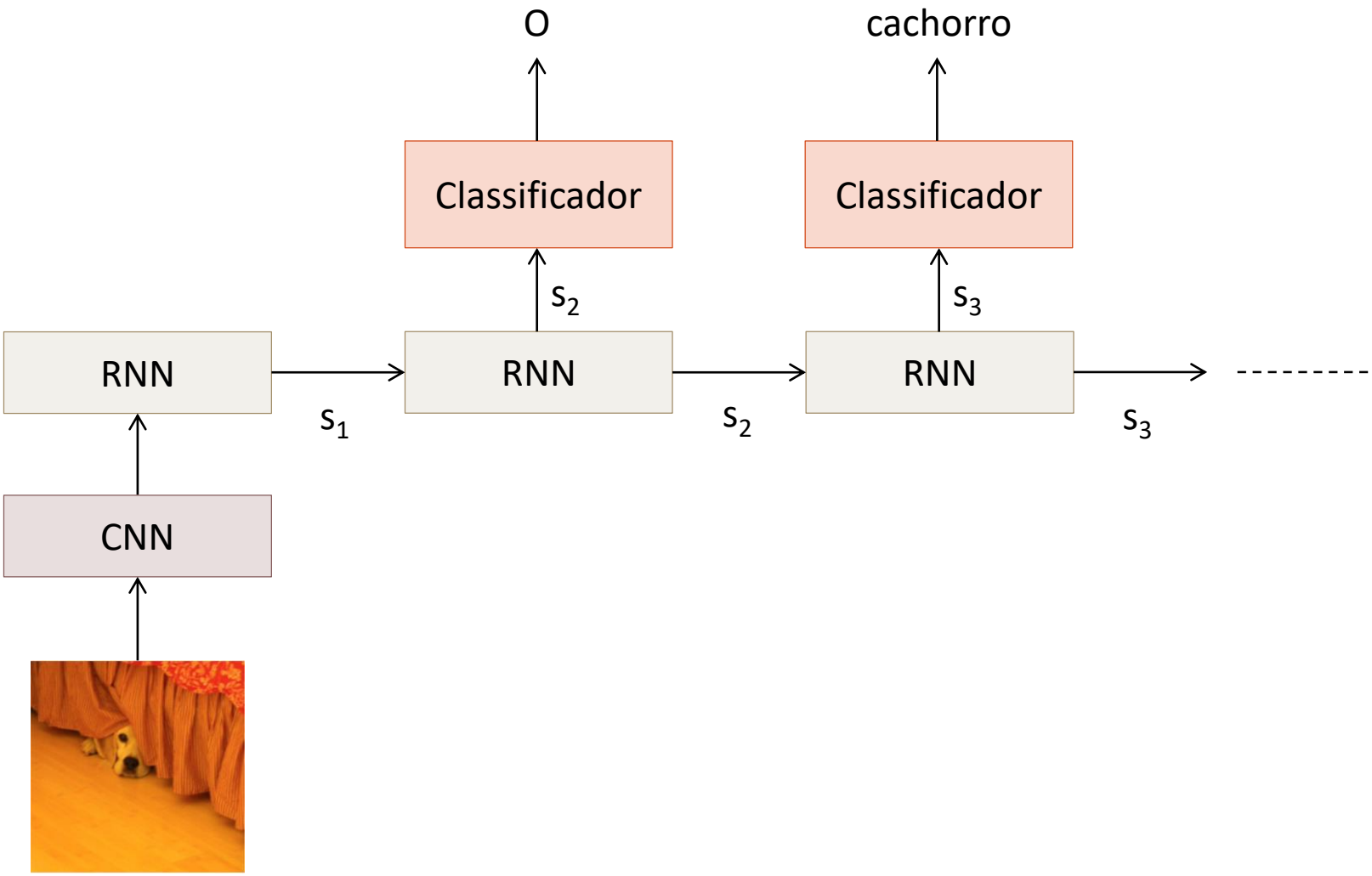
LEGENDAGEM DE IMAGEM



LEGENDAGEM DE IMAGEM



LEGENDAGEM DE IMAGEM



EXEMPLOS DE LEGENDAS DE IMAGEM

A person riding a motorcycle on a dirt road.

Two dogs play in the grass.

A herd of elephants walking across a dry grass field.

A group of young people playing a game of frisbee.

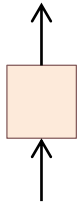
Two hockey players are fighting over the puck.

A close up of a cat laying on a couch.

Show and Tell: A Neural Image Caption Generator, CVPR 15

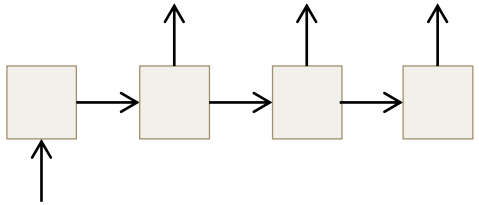
CENÁRIOS DE ENTRADA – SAÍDA

Simples - Simples



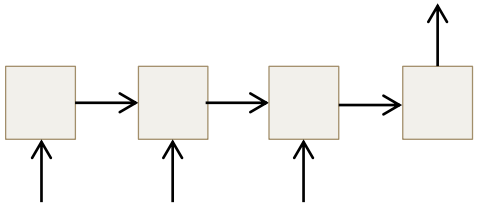
Rede Neural Feed-forward

Simples - **Múltiplo**



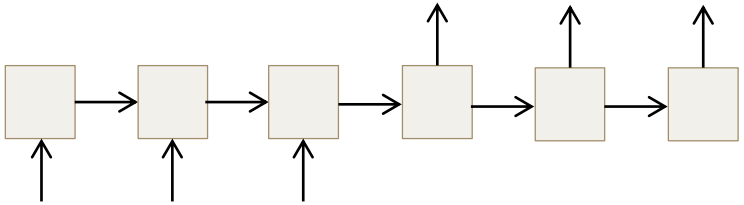
Legendagem de imagem

Múltiplo - Simples

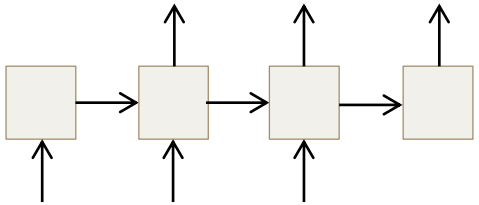


Classificação de sentimentos

Múltiplo - Múltiplo



Tradução



Legendagem de imagens

EXEMPLO DE APLICAÇÃO

Chegar Recife 2 Novembro

outro destino data date

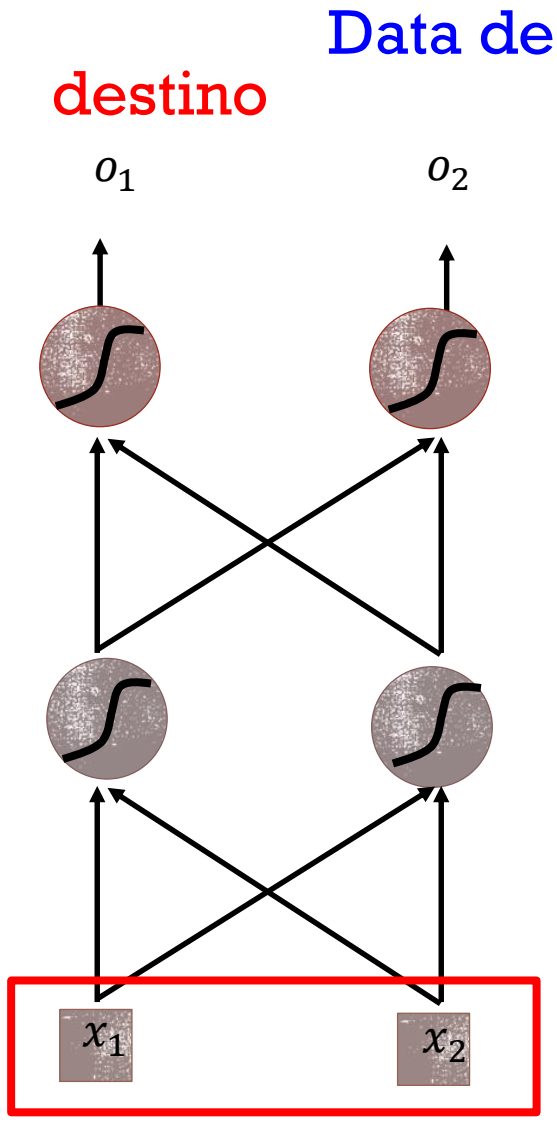
Problema?

Sair Recife 2 Novembro

Local de saída

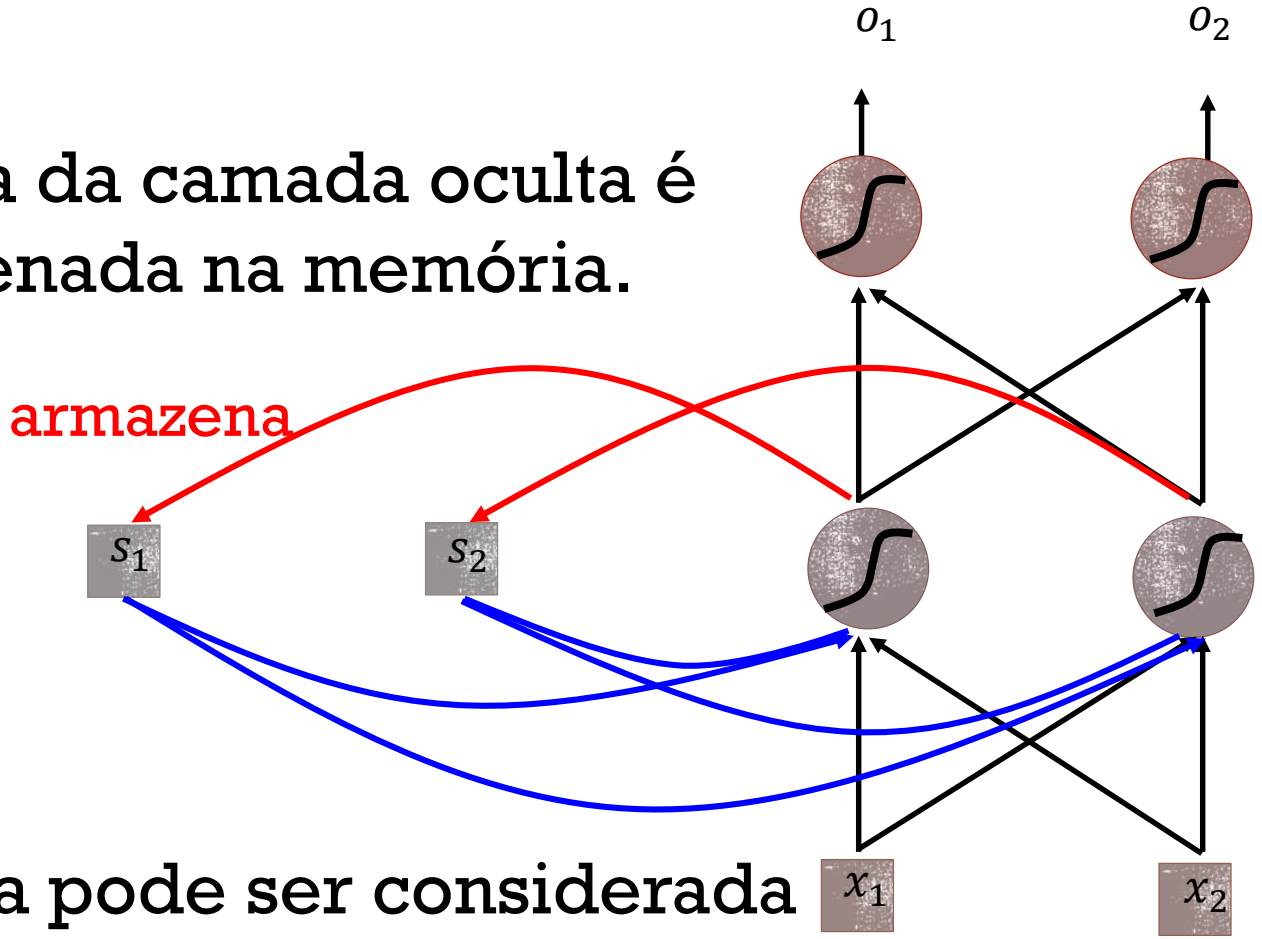
Precisa de memória

Recife



RECURRENT NEURAL NETWORK (RNN)

A saída da camada oculta é armazenada na memória.

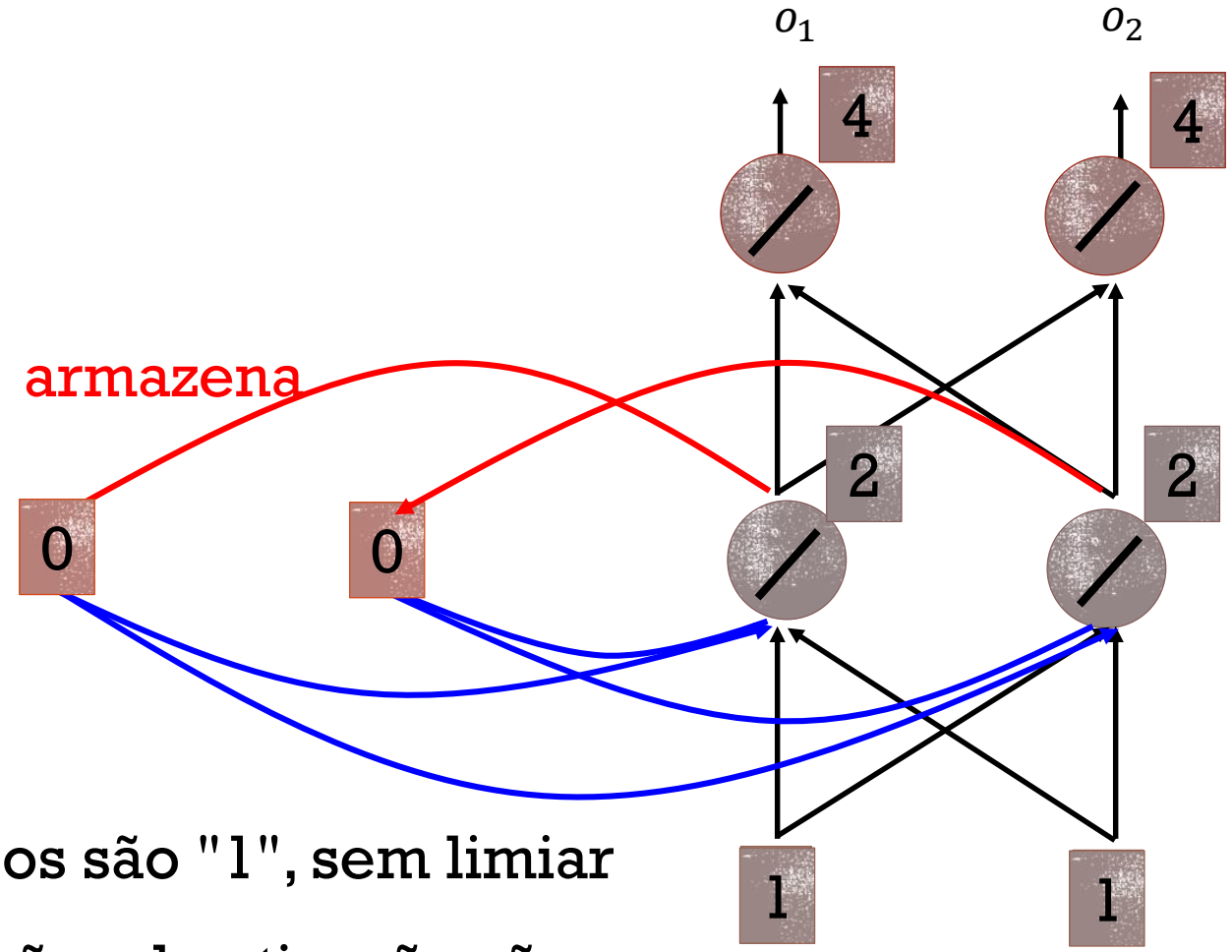


Memória pode ser considerada como outra entrada.

EXEMPLO

Sequência de entrada: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}2\\2\end{bmatrix}$ ... ...

Saída: $\begin{bmatrix}4\\4\end{bmatrix}$



Dados os valores iniciais

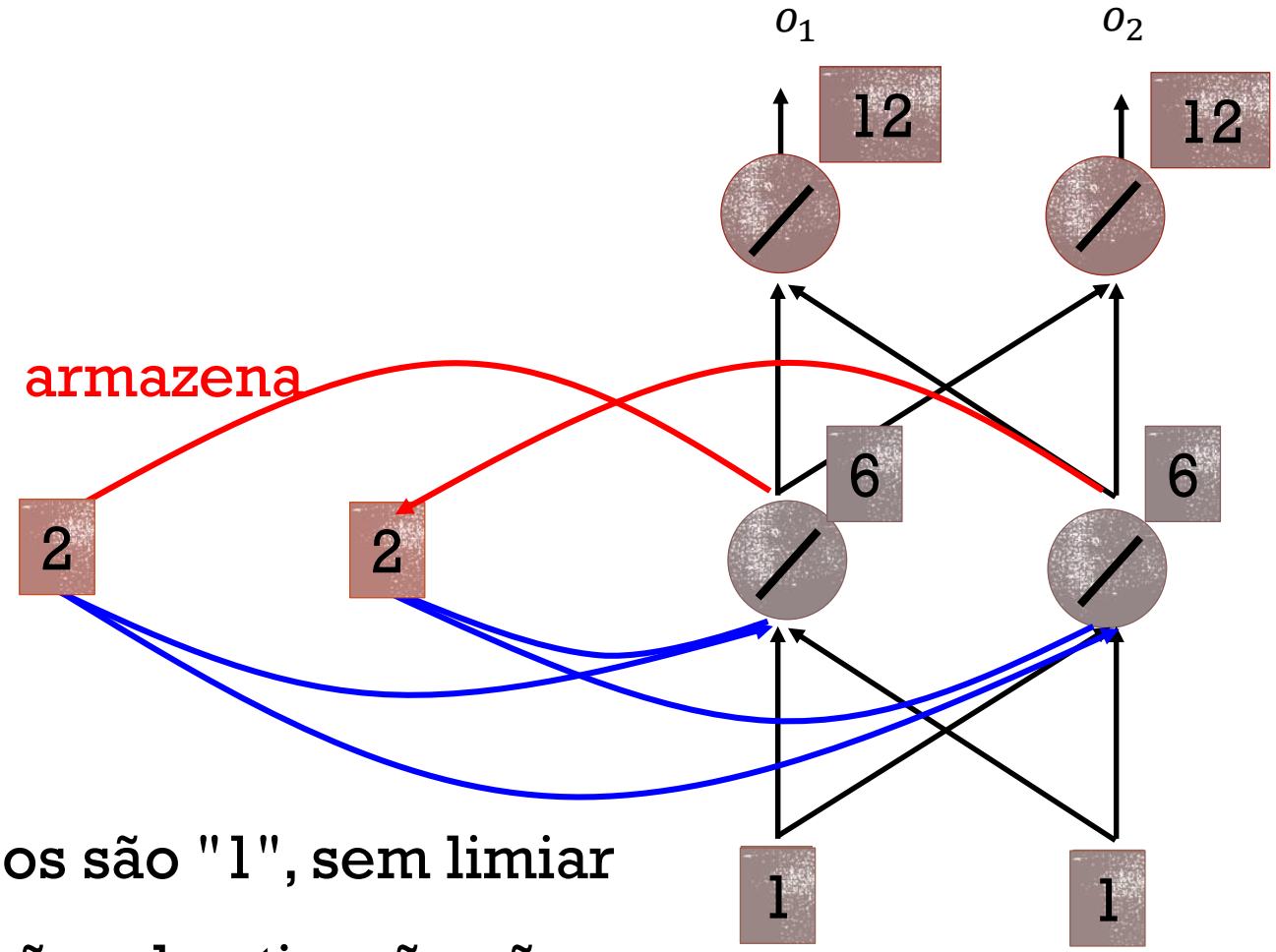
Todos os pesos são "1", sem limiar

Todas as funções de ativação são lineares

EXEMPLO

Sequência de entrada: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}2\\2\end{bmatrix}$ ... ...

Saída: $\begin{bmatrix}4\\4\end{bmatrix}$ $\begin{bmatrix}12\\12\end{bmatrix}$



Todos os pesos são "1", sem limiar

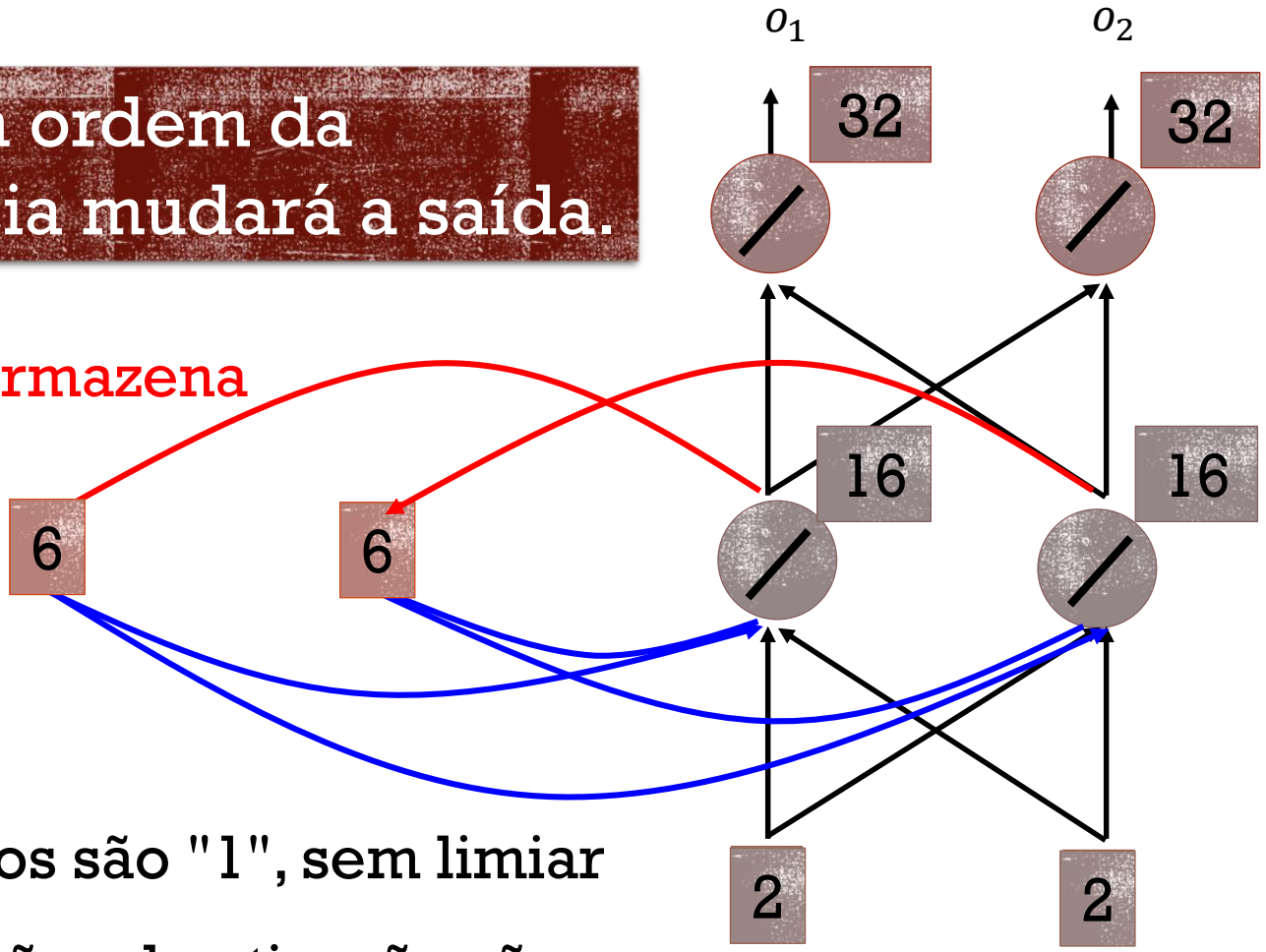
Todas as funções de ativação são lineares

EXEMPLO

Sequência de entrada: $\begin{bmatrix}1\\1\end{bmatrix}\ \begin{bmatrix}1\\1\end{bmatrix}\ \begin{bmatrix}2\\2\end{bmatrix}\ \ldots\ \ldots$

Saída $\begin{bmatrix}4\\4\end{bmatrix}\begin{bmatrix}12\\12\end{bmatrix}\begin{bmatrix}32\\32\end{bmatrix}$

Alterar a ordem da sequência mudará a saída.



Todos os pesos são "1", sem limiar

Todas as funções de ativação são lineares

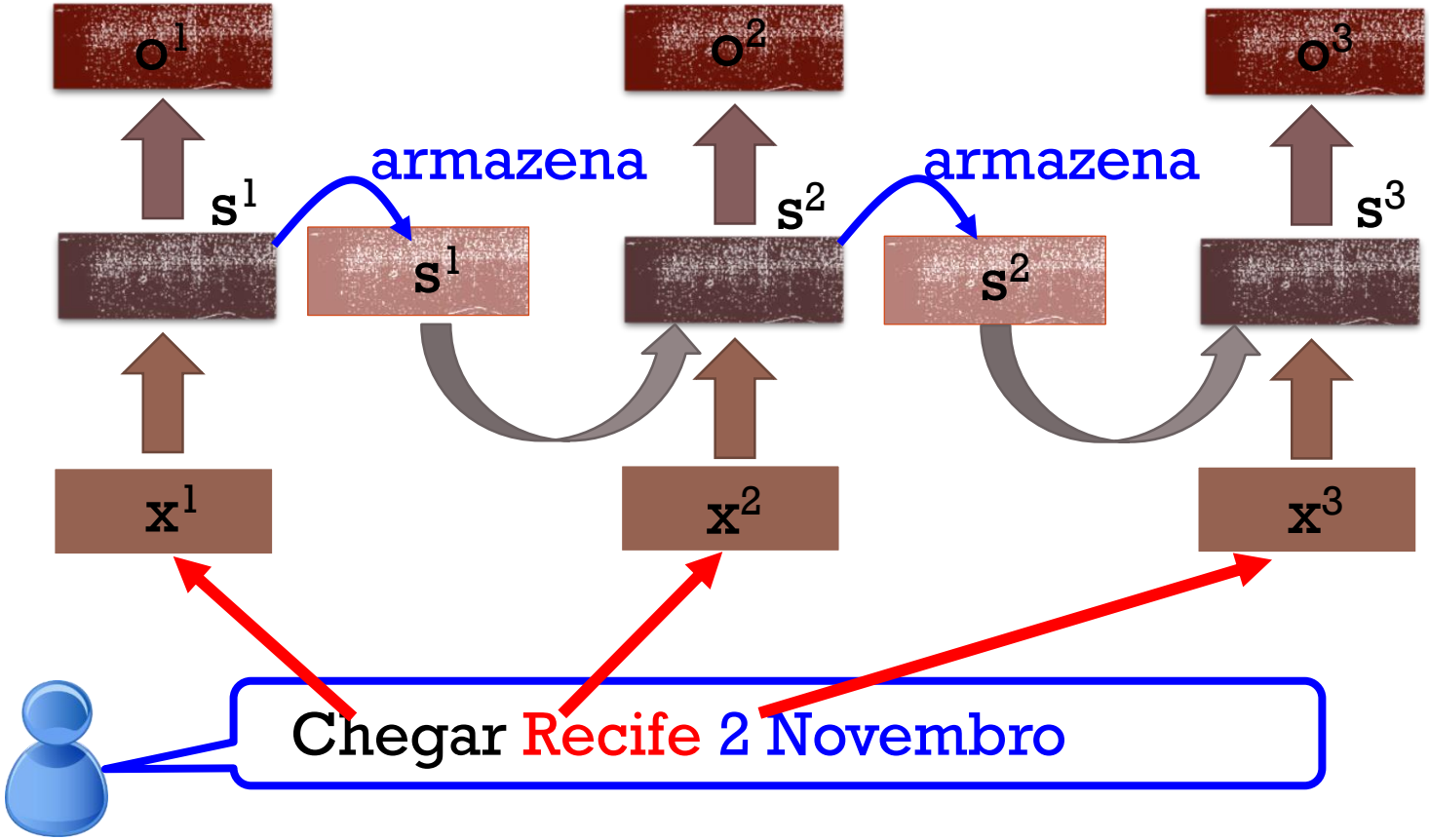
RNN

A mesma rede neural é usada repetidas vezes

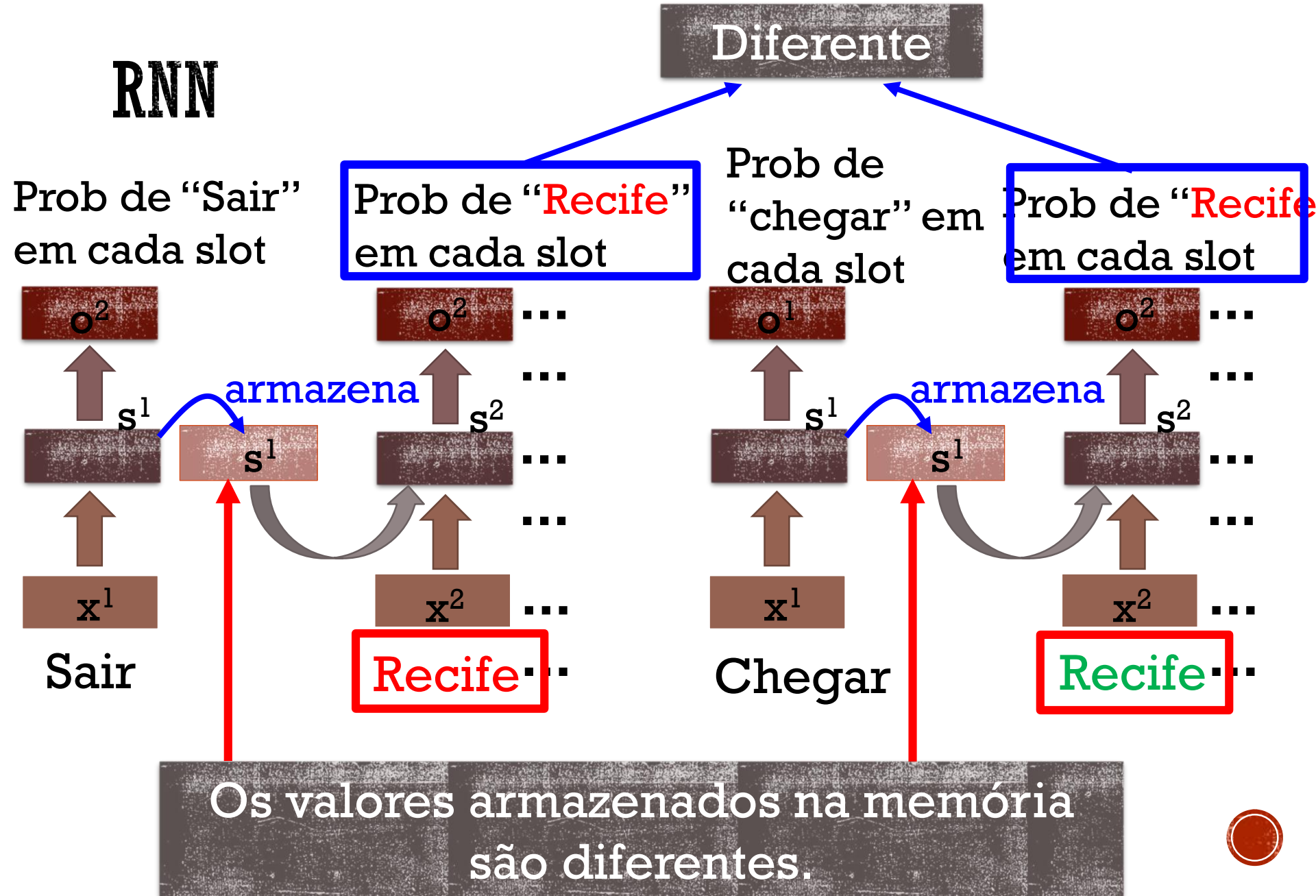
Probabilidade de "chegar" em cada slot

Probabilidade de "Recife" em cada slot

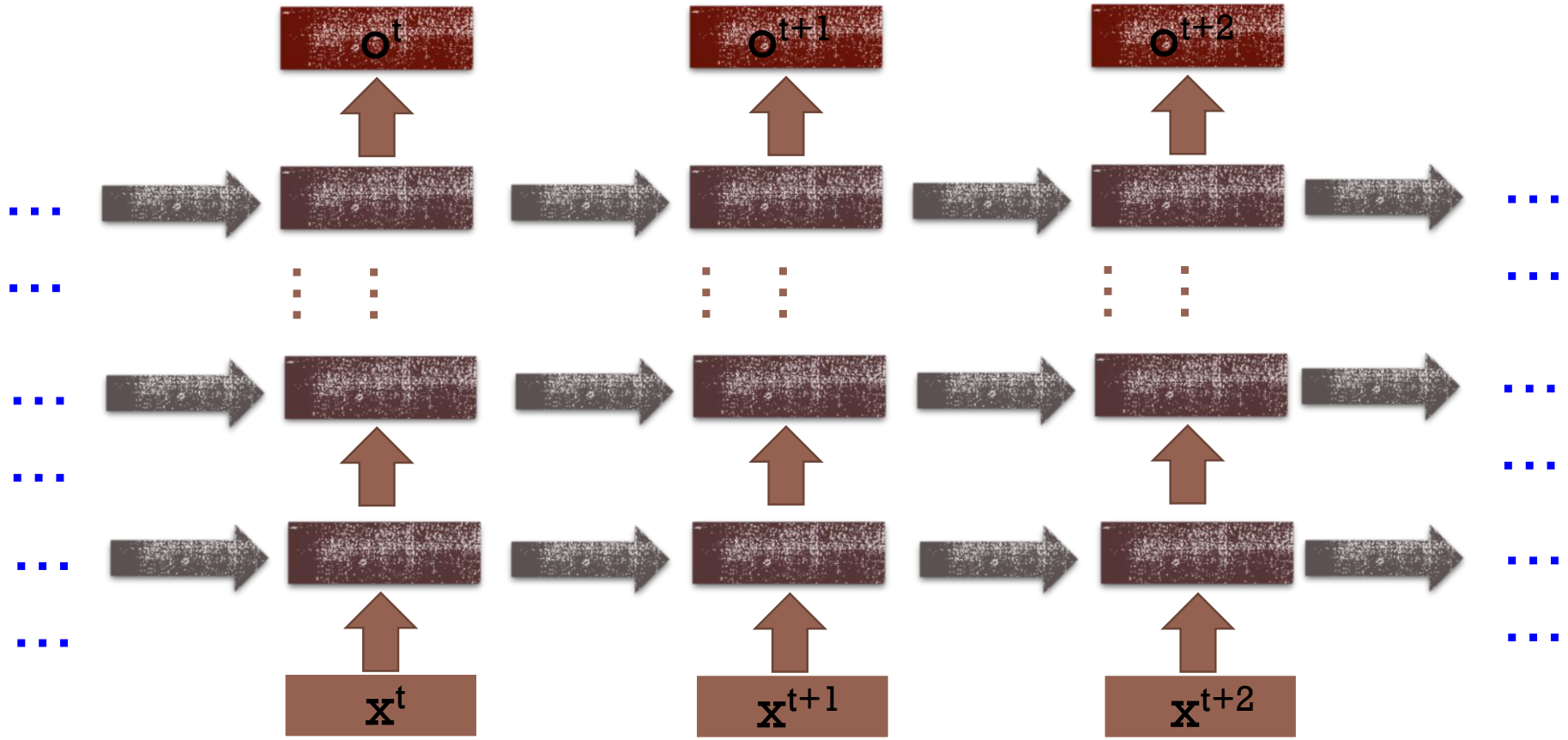
Probabilidade de "2" em cada slot



RNN


Diferente
Prob de "Sair" em cada slot
Prob de "Recife" em cada slot
Prob de "chegar" em cada slot
Prob de "Recife" em cada slot
armazena
armazena
Sair
Recife
Chegar
Recife
Os valores armazenados na memória são diferentes.

CLARO QUE PODE SER DEEP ...

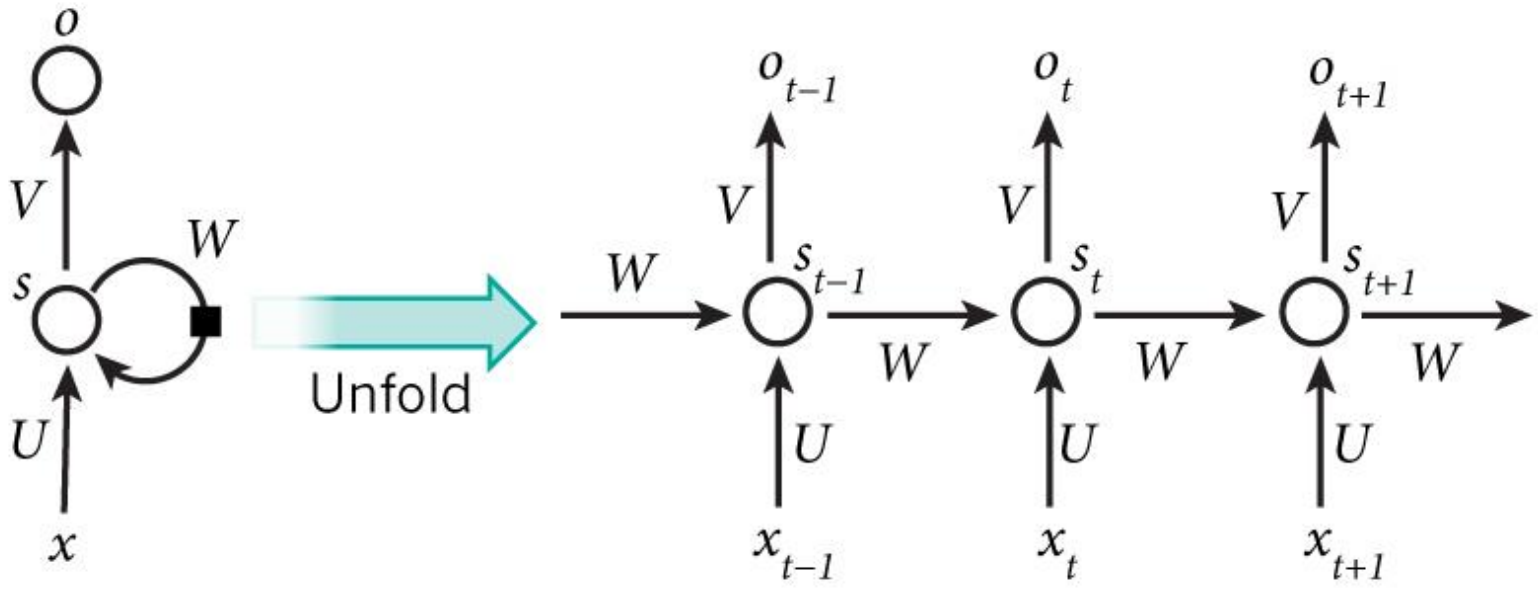


VANTAGENS E DESVANTAGENS

- Vantagens da RNN:
  - Pode processar qualquer comprimento de entrada
  - O tamanho do modelo não aumenta com uma entrada longa
  - Computação no passo *t* pode (em teoria) usar informações de muitos passos anteriores
  - Os pesos são compartilhados entre os tempos → as representações são compartilhadas
- Desvantagens da RNN:
  - A computação recorrente é lenta
  - Na prática, é difícil acessar informações de muitos passos anteriores

TREINAMENTO DA RNN

ARQUITETURA RNN



- Essa rede pode ser visualizada como uma rede neural feed forward desdobrando ao longo do tempo

TREINAMENTO DA RNN

Objetivo: obter os parâmetros de rede que otimizam a função de custo.
Funções de custo: perda logarítmica, erro médio quadrático, etc ...

Métodos:

Backpropagation:

- Convergência confiável e controlada
- Suportado pela maioria dos frameworks ML

Métodos evolutivos, maximização de expectativas, métodos não paramétricos, otimização de enxame de partículas → Pesquisa

RNN FORWARD

- Entrada da rede no tempo t:

$$h(t) = Ux(t) + Ws(t-1)$$

- Função de ativação da entrada no tempo t:

$$s(t) = \sigma(h(t))$$

- Entrada de rede para a unidade de saída no tempo t:

$$a_o(t) = Vs(t)$$

- A saída da rede no tempo t :

$$o(t) = \sigma\big(a_o(t)\big)$$

ARQUITETURA RNN

- Se uma sequência de treinamento começa no tempo t_0 e termina no tempo t_1, a **função de perda** total é a soma ao longo do tempo do erro quadrático

$$[e(t)]_k = [d(t)]_k - [o(t)]_k$$

$$E(t) = \frac{1}{2} e(t)^T e(t)$$

$$E_{total} = \sum_t E(t)$$

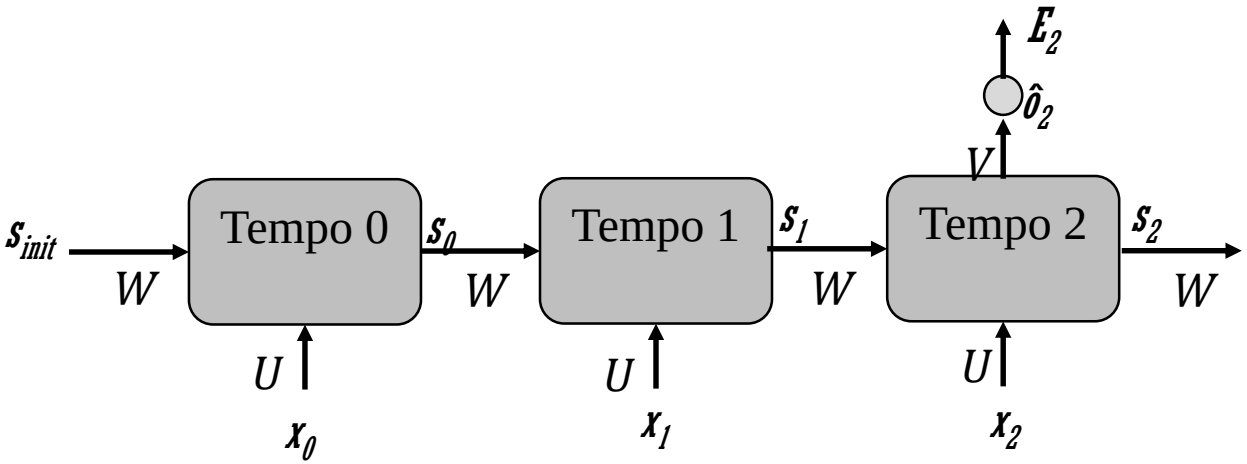
BACK-PROPAGATION THROUGH TIME

$$s(t) = \sigma(Ux(t) + Ws(t-1) + b)$$
$$o(t) = \sigma(Vs(t))$$

Perda da Entropia Cruzada: $E(\hat{o}, o) = -\sum_t o \log \hat{o}_t$

1. *Desdobre a rede*
2. *Repita para os dados de treinamento*:
    1. Dada uma sequência de entrada $\boldsymbol{x}$
    2. *Para t em* 0, *N*-1*:*
        1. *Passo Forward*
        2. Inicialize o estado intermediário para o valor anterior $s(t-1)$
        3. Obtenha a sequência de saída *o*
        4. Calcule o erro $E(\hat{o}, o)$
        5. *Back-propagate* o erro através do modelo desdobrado
        6. Média dos pesos
        7. Calcule o próximo estado intermediário $s(t)$

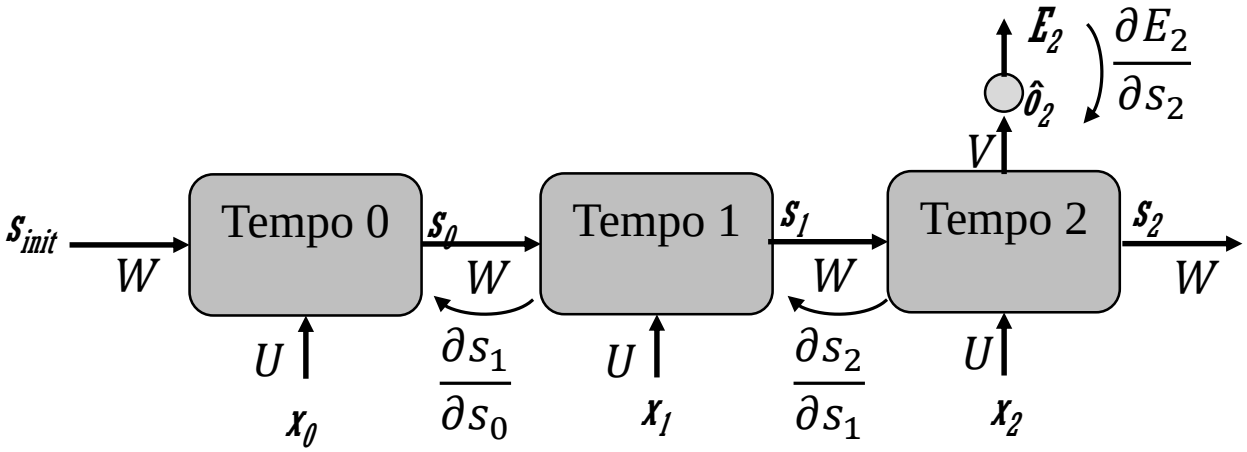
BACK-PROPAGATION THROUGH TIME (BPTT)



Aplicando a regra da cadeia:

$$\frac{\partial E_2}{\partial V} = \frac{\partial E_2}{\partial \hat{o}_2}\frac{\partial \hat{o}_2}{\partial V}$$

BACK-PROPAGATION THROUGH TIME (BPTT)



$$\frac{\partial E_2}{\partial W} = \frac{\partial E_2}{\partial \hat{o}_2}\frac{\partial \hat{o}_2}{\partial s_2}\frac{\partial s_2}{\partial W} + \frac{\partial E_2}{\partial \hat{o}_2}\frac{\partial \hat{o}_2}{\partial s_2}\frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial W} + \frac{\partial E_2}{\partial \hat{o}_2}\frac{\partial \hat{o}_2}{\partial s_2}\frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial s_0}\frac{\partial s_0}{\partial W}$$

$$\frac{\partial E_2}{\partial U} = \frac{\partial E_2}{\partial \hat{o}_2}\frac{\partial \hat{o}_2}{\partial s_2}\frac{\partial s_2}{\partial U} + \frac{\partial E_2}{\partial \hat{o}_2}\frac{\partial \hat{o}_2}{\partial s_2}\frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial U} + \frac{\partial E_2}{\partial \hat{o}_2}\frac{\partial \hat{o}_2}{\partial s_2}\frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial s_0}\frac{\partial s_0}{\partial U}$$

Para o tempo 2: $\frac{\partial E_2}{\partial \theta} = \sum_{k=0}^{2} \frac{\partial E_2}{\partial \hat{o}_2} \cdot \frac{\partial \hat{o}_2}{\partial s_0} \cdot \frac{\partial s_2}{\partial s_k} \cdot \frac{\partial s_k}{\partial \theta}$ $\boldsymbol{\theta}$ - Parâmetros (U, W)

BACK-PROPAGATION THROUGH TIME (BPTT)

- Formulação geral da derivada do erro

$$\frac{\partial E(t)}{\partial \theta} = \sum_{k=0}^{N} \frac{\partial E(t)}{\partial o_k(t)} \frac{\partial o_k(t)}{\partial \theta} = \sum_{k=0}^{N} e_k(t) \frac{\partial o_k(t)}{\partial \theta}$$

- Então:

$$\frac{\partial o_i(t+1)}{\partial \theta_{ij}} = f_i'\left(net_i(t)\right) \left[\sum_{k=0}^{N} \sum_{l=0}^{L} \theta_{kl} \frac{\partial z_l(t)}{\partial \theta} + \delta_k z_j(t) \right]$$

$$z_k(t) = \begin{cases} x_k(t) & if\ k \in U \\ w_k(t) & if\ k \in W \end{cases}$$

BACK-PROPAGATION THROUGH TIME (BPTT)

onde δ_{ij} é o delta de Kronecker

$$\delta_{ij} = \begin{cases} 1 & se \;\; i = j \\ 0 & se \;\; i \neq \mathrm{j} \end{cases}$$

- Como os sinais de entrada não dependem dos pesos da rede, equação superior se torna

$$\frac{\partial o_i(t+1)}{\partial \theta_{ij}} = f_i'\left(net_i(t)\right)\left[\sum_{k=0}^{N}\sum_{l=0}^{L}\theta_{kl}\frac{\partial o_l(t)}{\partial \theta} + \delta_k z_j(t)\right]$$

BACK-PROPAGATION THROUGH TIME (BPTT)

- Porque assumimos que o nosso estado inicial (t = 0) é independente dos pesos, temos

$$\frac{\partial o_k(t_0)}{\partial \theta_{ij}} = 0$$

- Portanto, precisamos definir os valores de sensibilidade

$$p_{ij}^k(t) = \frac{\partial o_k(t)}{\partial \theta_{ij}}$$

para cada passo de tempo e todos os apropriados i, j e k. Nós começamos com a condição inicial

$$p_{ij}^k(0) = 0$$

BACK-PROPAGATION THROUGH TIME (BPTT)

… e calcula pra cada tempo t

$$p_{ij}^{k}(t+1) = f_k'\left(net_k(t)\right)\left[\sum_{k=0}^{N}\sum_{l=0}^{L} o_{kl}\, p_{ij}^{l}(t) + \delta_k z_j(t)\right]$$

- O algoritmo consiste em calcular os valores sensitivos e, em seguida, usar o gradiente do peso

$$\Delta\theta_{ij}(t) = n\sum_{k=0} e_k(t)\, p_{ij}^{k}(t)$$

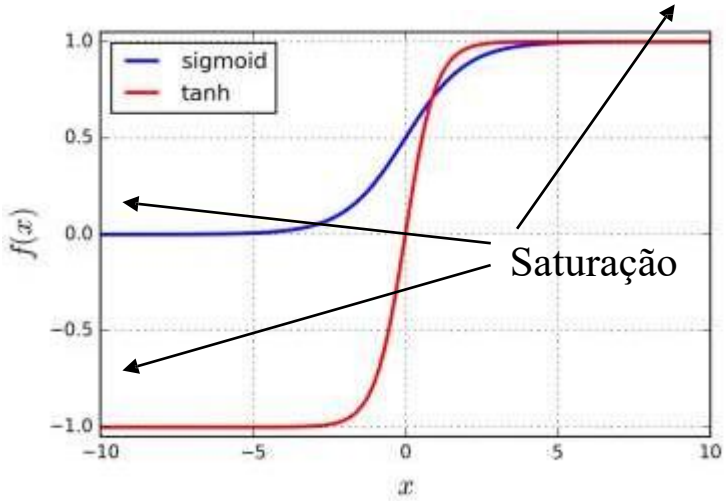
e a correção total de θ_{ij} é dada por

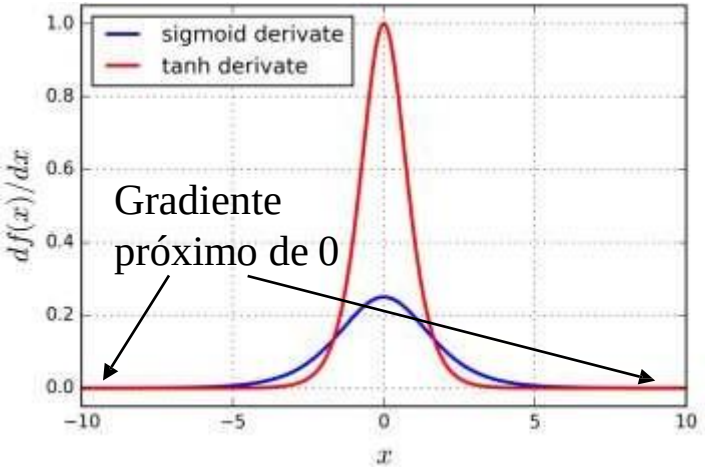
$$\Delta\theta_{ij}(t) = \sum_{k=k_0+1}^{k_1} \Delta\theta_{ij}(t)$$

DISSIPAÇÃO/EXPLOSÃO DOS GRADIENTES

- Dois problemas que podem ocorrer durante o treinamento de RNNs:
  - *Vanishing gradients*
  - *Exploding gradients*
- Uma das causas: o fluxo de informação (*propagação dos gradientes*) em uma RNN ocorre por meio de sequências de multiplicações.

PROBLEMA: DISSIPAÇÃO DO GRADIENTE





Gradientes de neurônios saturados → 0

$$\frac{\partial h_t}{\partial h_0} = \frac{\partial h_t}{\partial h_{t-1}} \cdots \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial h_0}$$

- Comum se o espaço de tempo aumenta para *t* maior que 8 ou 10
- Ou se número de camadas aumenta

- Decai exponencialmente
- A rede para de aprender, não consegue atualizar
- Impossível aprender correlações entre eventos temporariamente distantes

DISSIPAÇÃO/EXPLOSÃO DOS GRADIENTES

- *Vanishing gradients*
  - Os gradientes "desaparecem" (i.e., tendem a zero) quando são continuamente multiplicados por números inferiores a um.
- *Exploding gradients*
  - *Os gradientes se tornam exponencialmente grandes quando são multiplicados por números maiores que 1.*

EXPLOSÃO DOS GRADIENTES *(GRADIENT EXPLOSION)*

- Algumas técnicas alternativas para atenuar o problema da dissipação\explosão dos gradientes:
  - Truncar o BPTT (*truncated BPTT*)
  - Truncar os gradientes (*gradient clipping*)
  - Usar otimizadores que ajustem adaptativamente a taxa de aprendizado (AdaDelta, RMSProp, Adam, ...).

TRUNCAR O BPTT (*TRUNCATED BACKPROPAGATION*)

- Corresponde a limitar a quantidade de passos de tempo no passo *backward* a um valor máximo preestabelecido.
  - e.g., aplicar backprop a cada 10 passos.
- Desvantagem: perda do contexto temporal.
  - "(*truncated backpropagation*)"

TRUNCAR OS GRADIENTES (*TRUNCATED GRADIENTS, GRADIENT CLIPPING*)

- Se o comprimento do vetor gradiente ultrapassar um valor máximo preestabelecido, truncar.
- Várias possibilidades...uma delas:

  new_gradients = gradients * threshold / l2_norm(gradients)
- Técnica simples e efetiva!

Algorithm 1 Pseudo-code for norm clipping the gradients whenever they explode

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if

Fonte: https://arxiv.org/pdf/1211.5063.pdf

DISSIPAÇÃO DOS GRADIENTES *(VANISHING GRADIENTS)*

- Muito mais difícil de detectar!
- Alternativas de solução:
  - Usar boas estratégias de inicialização dos pesos
  - Usar otimizadores que adaptam a taxa de aprendizado (e.g., RMSProp).
  - Usar regularização (L1/L2) na função de perda
  - Usar LSTMs ou GRUs