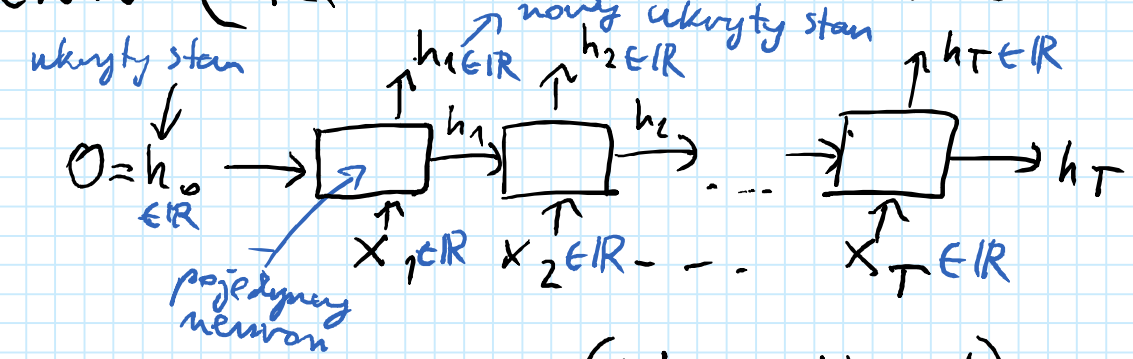
Uczenie maszynowe 7.12.2022

RNN (Recurrent Neural Network)



$$h_t = \sigma(Wx_t + Uh_{t-1} + b)$$

W, U, b – wagi

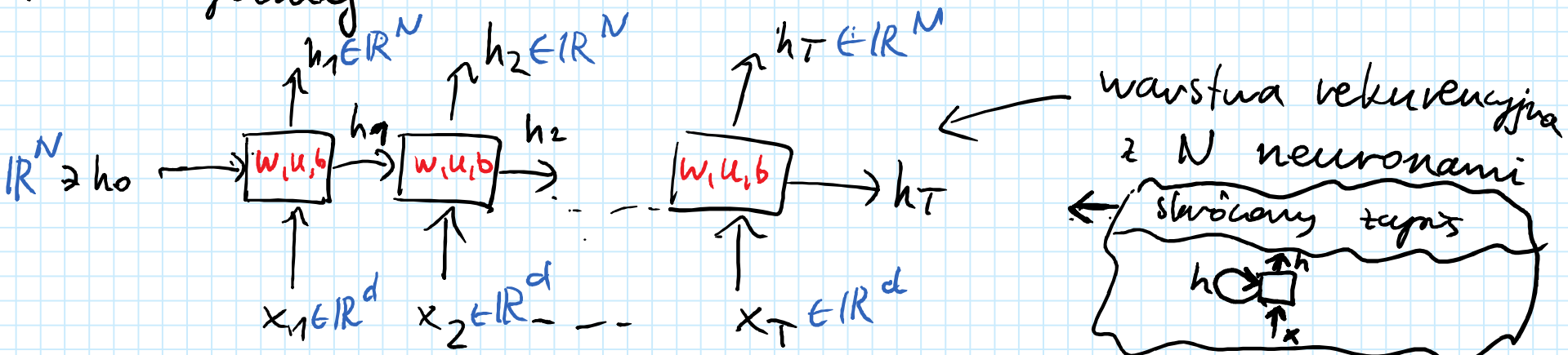
Zastosowania

- modelowanie szeregów czasowych
- przetwarzanie języka naturalnego

Przykłady

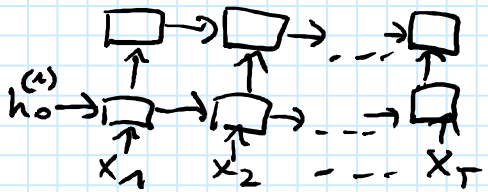
- prognozowanie cen zamknięcia akcji na podstawie danych historycznych
- klasyfikacja recenzji („gwiazdki")
- wykrywanie sarkazmu (/s na Reddicie) itd.

RNN – ogólniej



$$\underbrace{h_t}_{\in \mathbb{R}^N} = \sigma\left(\underbrace{W}_{\in \mathbb{R}^{N \times d}} \underbrace{x_t}_{\in \mathbb{R}^d} + \underbrace{U}_{\in \mathbb{R}^{N \times N}} \underbrace{h_{t-1}}_{\in \mathbb{R}^N} + \underbrace{b}_{\in \mathbb{R}^N} \right)$$

- σ nakładamy po współrzędnych
- W, U, b są takie same w każdym kroku czasowym
- pamięć jest współdzielona pomiędzy wszystkie neurony
- liczba wag $= Nd + N^2 + N$
- ostateczne wyjście z sieci to h_T albo $(h_1, h_2, \ldots, h_T)$

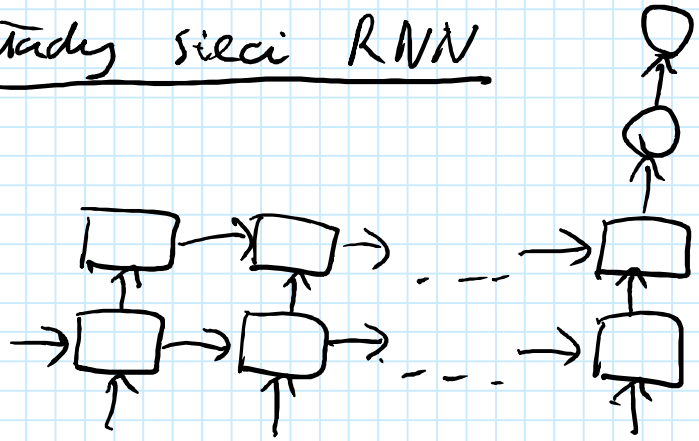


druga warstwa rekurencyjna

pierwsza warstwa rekurencyjna

Przykłady sieci RNN

1)

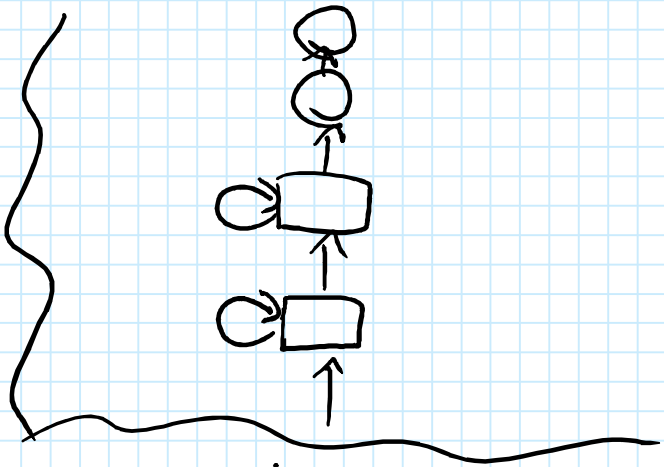


gęsta

gęsta

rekurencyjna

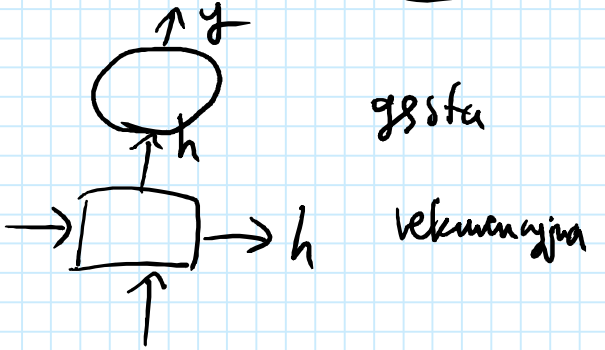
rekurencyjna



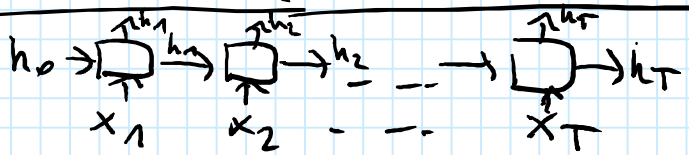
2) Elman Network

$$h_t = \sigma\left(W_h x_t + U_h h_{t-1} + b_h\right)$$

$$y_t = \sigma_y\left(W_y x_t + b_y\right)$$



PROBLEM ZNIKAJĄCYCH GRADIENTÓW

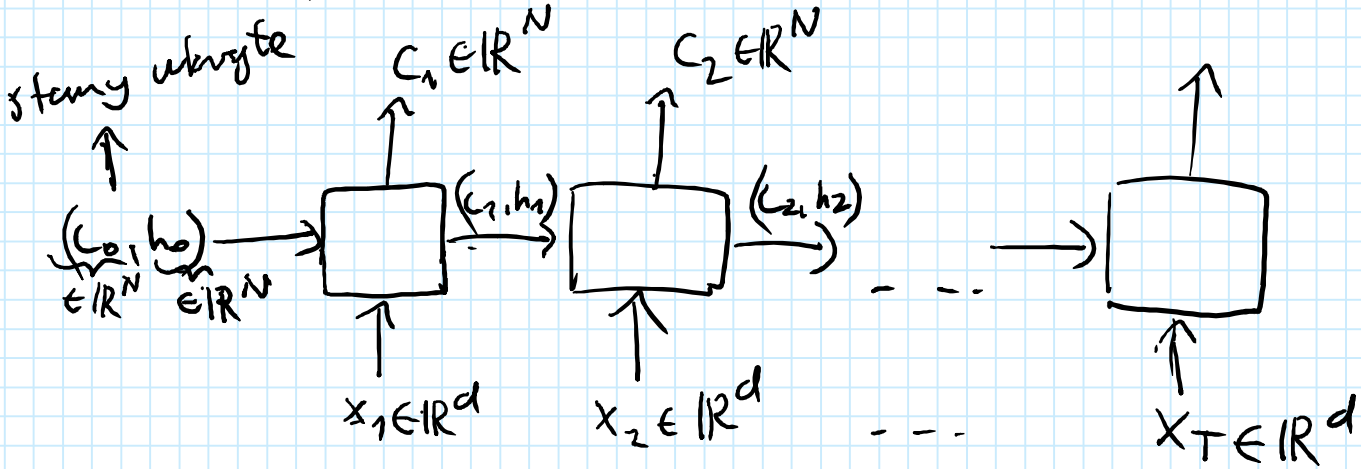


- Ostatni x_T przechodzi tylko przez jedną grupę neuronów ale x_1 przechodzi przez znacznie więcej, więc sieć jest efektywnie bardzo głęboka
- Jeśli T jest duże, to x_1 jest bardzo daleko od x_T i trudno mu „dojść do h_T".

Rozwiązanie: LSTM !

LSTM (Long Short-Term memory)

Hochreiter, Schmidthuber 1997



Liczba wag
$= 4(Nd + N^2 + N)$

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \in \mathbb{R}^N$$ forget gate

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$ input gate

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$ output gate

$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = o_t \odot \sigma_h(c_t)$$

σ_g – sigmoid

σ_c, σ_h – tanh

$\odot$ – mnożenie po współrzędnych (Hadamard product)

$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$

- $f_t^{(k)} \approx 1$, $i_t^{(k)} \approx 0$ $\Rightarrow$ propagujemy tę k-tą współrzędną dalej
- $f_t^{(k)} \approx 0$, $i_t^{(k)} \approx 1$ $\Rightarrow$ zapominamy tę k-tą współrzędną i zastępujemy ją nowym kandydatem $\tilde{c}$

GRU (Gated Recurrent Unit)

Kyunghyun Cho et al. „Learning Phrase Representation..." 2014

- mniej wag niż dla LSTM

podobne działanie do LSTM

- brak output gate

Liczba wag $= 3(Nd + N^2 + N)$

$$z_t = \sigma_g\left(\underbrace{W_z}_{\in \mathbb{R}^{N\times d}} \underbrace{x_t}_{\in \mathbb{R}^d} + \underbrace{U_z}_{\in \mathbb{R}^{N\times N}} \underbrace{h_{t-1}}_{\in \mathbb{R}^N} + \underbrace{b_z}_{\in \mathbb{R}^N}\right) \in \mathbb{R}^N \quad \text{update gate}$$

$$r_t = \sigma_g\left(W_r x_t + U_r h_{t-1} + b_r\right) \quad \text{reset gate}$$

$$\hat{h}_t = \sigma_h\left(W_h x_t + U_h\left(r_t \odot h_{t-1}\right) + b_h\right) \quad \text{candidate activation}$$

σ_g - sigmoid

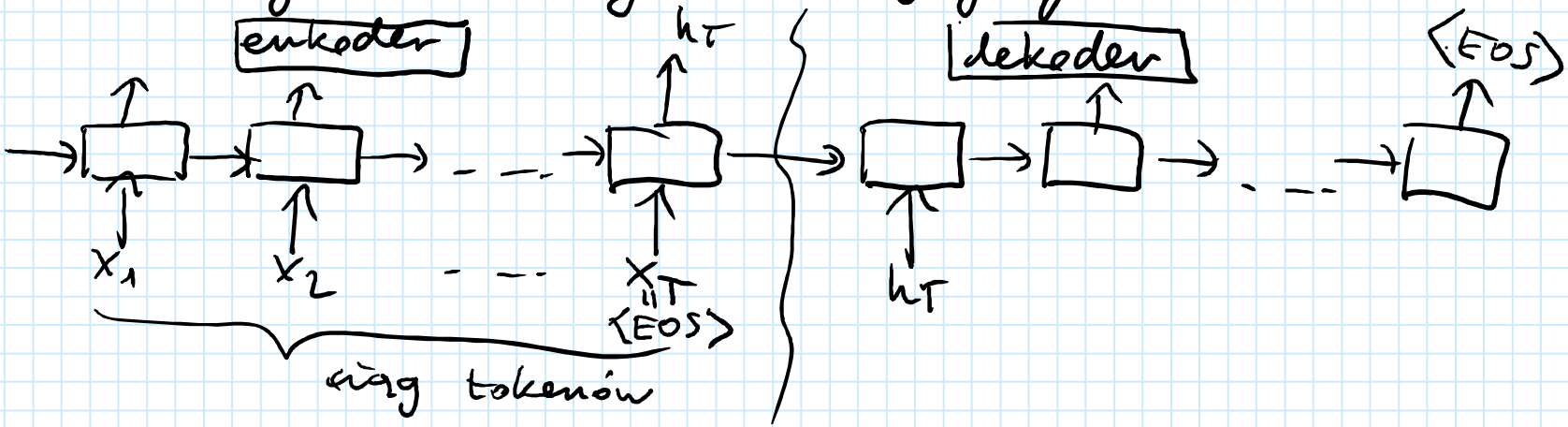
σ_h - tanh

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

- Jeśli $z_t \approx 0$ na pewnej współrzędnej, to na tej współrzędnej zostawiamy h_t bez zmian
- Jeśli $z_t \approx 1$ na pewnej współrzędnej i to na tej współrzędnej zapominamy przeszłość i zastępujemy ją nową $\hat{h}_t$

Architektura encoder – decoder

- są 2 części rekurencyjne: encoder oraz decoder
- encoder koduje zdanie zapisane w jednym języku, a decoder je odkodowuje na inny język.



$\begin{bmatrix} \vdots \\ \text{<UNK>} \\ < > \end{bmatrix}$ – specjalne słowa np. <UNK> – nieznany wyraz, którego nie ma w słowniku; <EOS> – end of sentence (koniec zdania)

- RepeatVector
- return_sequences = TRUE