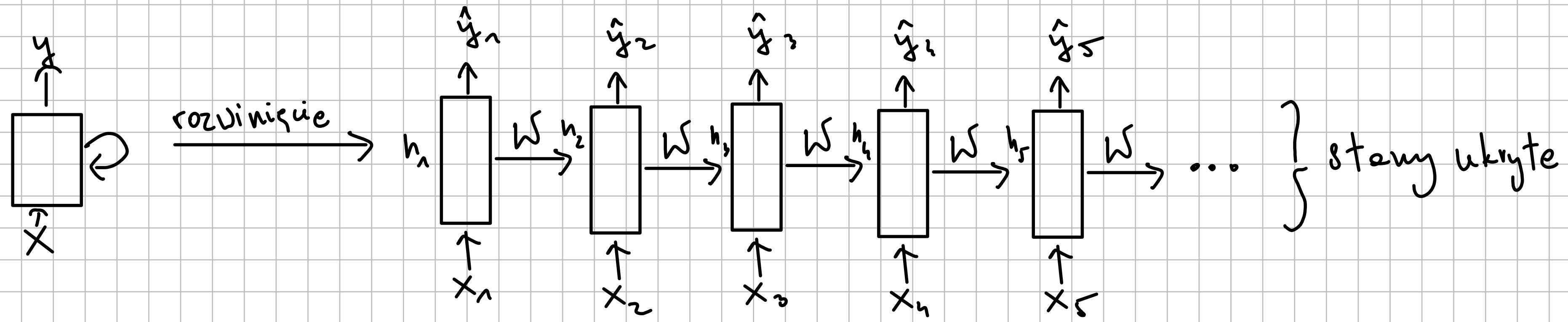
# RNN (Recurrent neural network)

- Rozwiązanie problemu wejścia o zmiennej długości
- Główna idea - użycie tej samej macierzy wag W w wielu krokach



# Bardziej dokładny schemat modelu (językowego)

$x_{(i)}$ - słowa, zakodowane jako one-hot-encoding

"word embedding"
[przedstawienie wektorowe]

$e_i = E x_i$

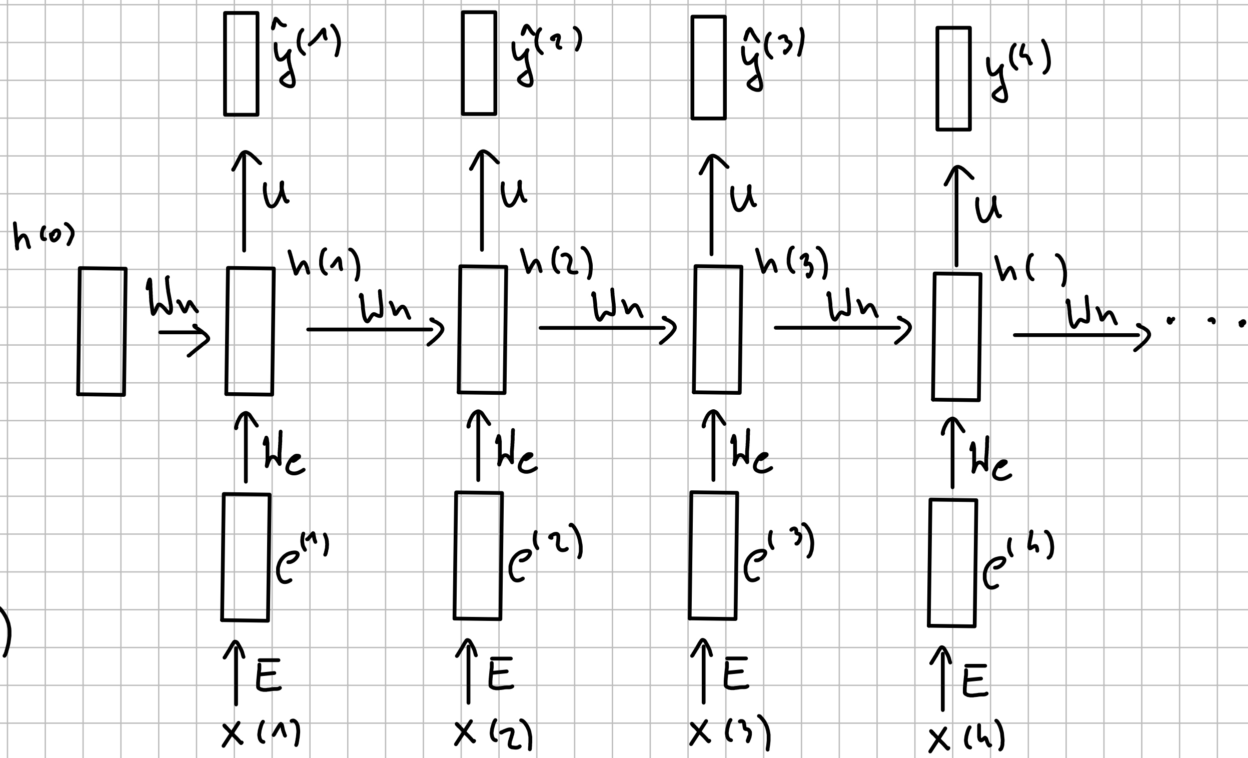
stany ukryte

$h(0)$ - stan początkowy, najczęściej wektor zer

$h(i) = \sigma(W_h h(i-1) + W_e e(i) + b_1)$

stany wyjściowe

$y(i) = \mathrm{softmax}(U h(i) + b_2)$



# Zalety i wady RNN

## Zalety:

- Wejście może być dowolnie długie
- W teorii w czasie $t$ może mieć informację z wielu kroków wcześniej
- Ta sama macierz $W$ używana w każdym kroku
    - → Ilość parametrów do trenowania nie zależy od długości wejścia
    - → stałość tego jak traktujemy wejście

## Wady:

- Część rekurencyjna sprawia że obliczenia są długie
- W praktyce w czasie $t$ model "nie pamięta" zbyt dobrze dalekiej przeszłości

# Trenowanie RNN

- Duże wejście → np. książka, tekst z wikipedii
- Przepuszczamy przez RNN i obliczamy $\hat{y}(i)$ dla każdego kroku
- Funkcja kosztu - Entropia krzyżowa

Dla jednego kroku:

$$J(t)(\theta) = CE\left(y(i) - \hat{y}(i)\right) = -\sum_{w \in V} \underbrace{y_w(t)}_{\text{one-hot}} \log \hat{y}_w(t) = -\log \hat{y}_{x(t+1)}(t)$$

Całościowy koszt:

$$J(\theta) = \frac{1}{T}\sum_{t=0}^{T} J(t)(\theta) = -\frac{1}{T}\sum_{t=0}^{T} \log \hat{y}_{x(t+1)}(t)$$

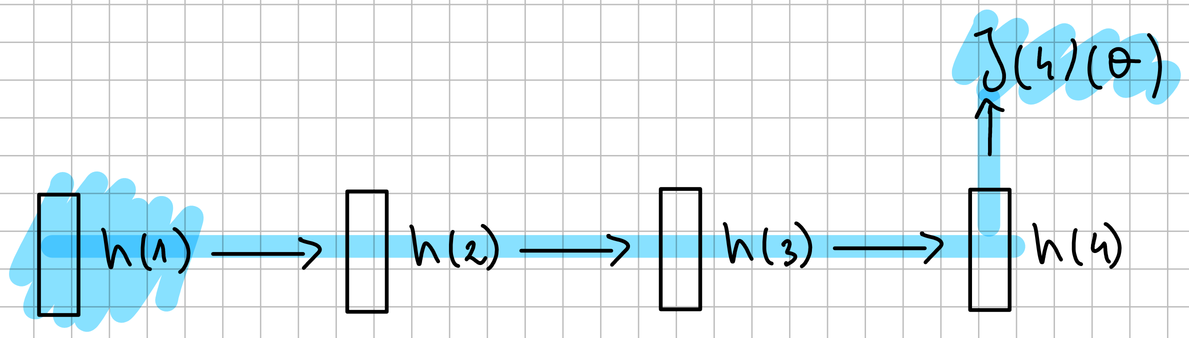
W praktyce nie przechodzimy przez cały tekst na raz, tylko obliczamy $J(\theta)$ dla mniejszego podzbioru (np 16 zdań), obliczamy gradienty aktualizujemy wagi i kontynuujemy

# Back propagacja

$$\frac{\partial J^{(t)}}{\partial W_h} = \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial W_h}\Big|_i$$ ← "Backpropagation through time"

- Problem → jak zdanie / tekst jest zbyt długie to trwałoby to długo!
- rozwiązanie → "ucięta" backpropagacja w czasie, przechodzimy w tył nie przez całe zdanie, ale przez z góry ustaloną liczbę kroków, np 20

# Problem znikającego/wybuchającego gradientu




Obliczamy gradient $J^{(4)}(\theta)$ względem $h^{(1)}$, przy pomocy metody łańcucha:

$$\frac{\partial J^{(4)}(\theta)}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \cdot \frac{\partial h^{(3)}}{\partial h^{(2)}} \cdot \frac{\partial h^{(4)}}{\partial h^{(3)}} \cdot \frac{\partial J^{(4)}(\theta)}{\partial h^{(4)}}$$

→ Wielokrotne mnożenie → małe liczby znikają do 0

→ duże liczby eksplodują

# Przykład dla $\sigma(x) = x$ (funkcja identycznościowa)

$$h(t) = \sigma\left(W_h h(t-1) + W_x x(t) + b_1\right)$$

$$\frac{\partial h(t)}{\partial h(t-1)} = \operatorname{diag}\left(\sigma'\left(W_h h(t-1) + W_x x(t) + b_1\right)\right) W_h = I W_h = W_h$$

$$\Downarrow$$

$$\frac{\partial J(i)}{\partial h(j)} = \frac{\partial J(i)}{\partial h(i)} \cdot \prod_{j<t\le i} \frac{\partial h(t)}{\partial h(t-1)} = \frac{\partial J(i)}{\partial h(j)} \cdot (W_h)^l \quad [i-j=l]$$

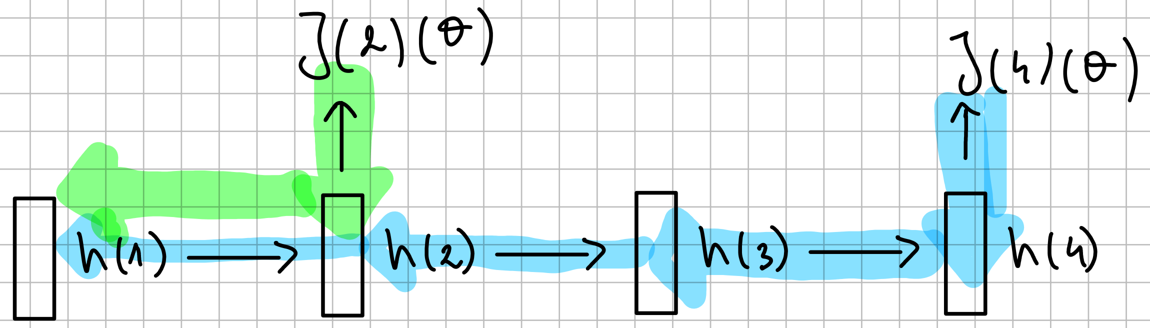
↑ może maleć wykładniczo

dla $W_h$ t.że wartości własne $\lambda_1, \dots, \lambda_n < 1$: [warunek dostateczny ale nie konieczny]

$$\frac{\partial J(i)}{\partial h(j)} (W_h)^l = \sum_{i=1}^{n} c_i (\lambda_i)^l q_i \approx 0 \text{ dla dużych } l$$

[Dla bardziej skomplikowanych $\sigma$ dzieje się to samo tylko dla $\lambda_i < \gamma$ które zależy od tego jakie wybierzemy $\sigma$]

# Skutki znikającego gradientu




- Wpływu odległego sygnału znika bo rzędy wielkości gradientów mogą być zupełnie różne
- Model nie potrafi "pamiętać" informacji z dalekiej przeszłości

# Przykład:

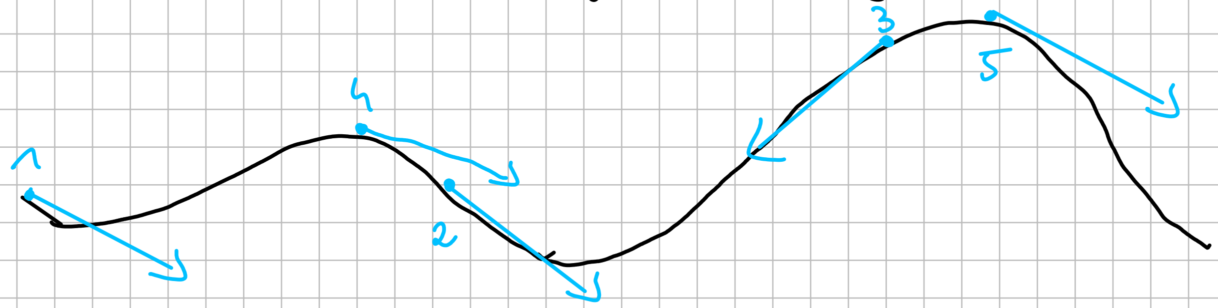
"Kiedy Asia chciała wydrukować bilet, okazało się że nie ma tuszu w drukarce. Asia poszła do sklepu kupić toner, był akurat na przecenie. Włożyła nowy toner do drukarki i wydrukowała ______"

25 kroków różnicy! → znikający gradient sprawia, że wpływ słowa "bilet" z tak dawna jest znikomy i nie potrafimy "zapamiętać" informacji z dalekiej przeszłości

# Skutki eksplodującego gradientu

$$\theta^{new} = \theta^{old} + \alpha \underbrace{\nabla_\theta J(\theta)}_{\text{gradient}}$$

- Jak gradient jest zbyt duży, to robimy zbyt duże kroki i błądzimy



# Rozwiązanie problemu eksplodującego gradientu

- "Zmniejszenie kroku przy zachowaniu kierunku chodzenia"

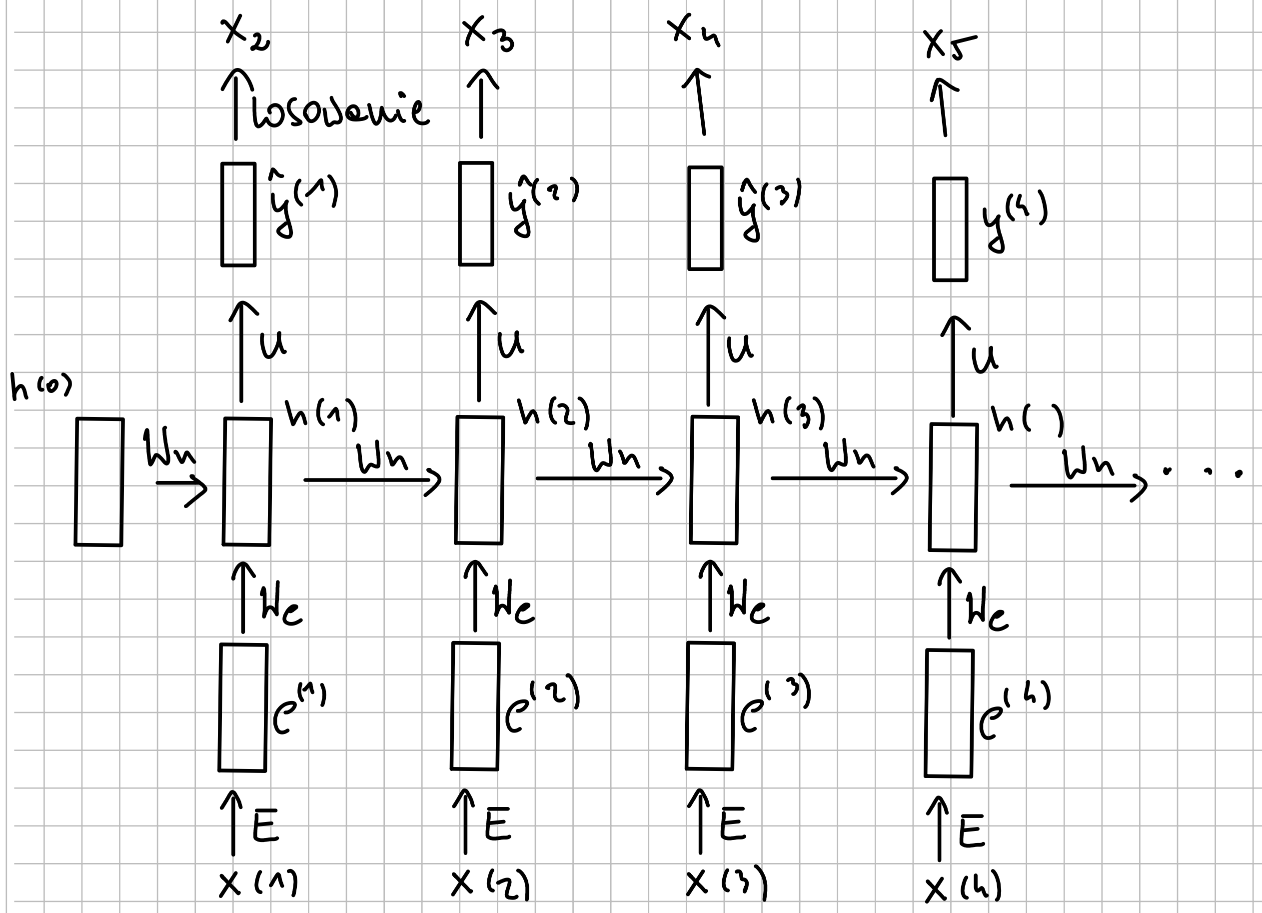
Pseudokod:

- $\hat{g} \leftarrow \frac{\partial \varepsilon}{\partial \theta}$
- if $\|\hat{g}\| > \tau$ then

  $\hat{g} \leftarrow \frac{\tau}{\|\hat{g}\|} \hat{g}$

  end if

# Generowanie tekstu przez RNN




Wartość wylosowaną w 1. kroku traktujemy jako 2. element wejścia itd.