# Neuronowe modele sekwencyjne

## Zaawansowane Przetwarzanie Języka Naturalnego

Mateusz Lango

Zakład Inteligentnych Systemów Wspomagania Decyzji
Wydział Informatyki i Telekomunikacji
Politechnika Poznańska

„Akademia Innowacyjnych Zastosowań Technologii Cyfrowych (AI Tech)",
projekt finansowany ze środków Programu Operacyjnego Polska Cyfrowa POPC.03.02.00-00-0001/20

Fundusze Europejskie
Polska Cyfrowa

Rzeczpospolita Polska

Unia Europejska
Europejski Fundusz Rozwoju Regionalnego

# Problem predykcji sekwencji/struktur

- Ukryte Modele Markowa

$$P(x_1^n, y_1^n) = \prod_{i=1}^{n} P(x_i|y_i) \prod_{i=1}^{n+1} P(y_i|y_{i-1})$$

- Modele Markowa o Maksymalnej Entropii (MEMM)

$$P(y_1^n|x_1^n) = \prod_{i=1}^{n} P(y_i|y_{i-1}, x_1^n) = \prod_{i=1}^{n} \frac{e^{\mathrm{w}^T \phi(\mathrm{x}_i, y_{i-1}, y_i)}}{\sum_{y'} e^{\mathrm{w}^T \phi(\mathrm{x}_i, y_{i-1}, y')}}$$

- Warunkowe pola losowe (CRF)

$$P(\mathrm{y}|\mathrm{x}) = \frac{e^{\sum_{i=1}^{n} \mathrm{w}^T \phi(y_i, y_{i-1}, \mathrm{x})}}{\sum_{\mathrm{y}'} \prod_{i=1}^{n} e^{\mathrm{w}^T \phi(y_i', y_{i-1}', \mathrm{x})}}$$

# Ograniczenia Ukrytych Modeli Markova[1]

- HMM przyjmują jeden z $|C|$ stanów ukrytych (np. po jednym dla każdej części mowy), które zmieniają się zgodnie z prawdopodobieństwami tranzycji $P(y_t|y_{t-1})$ a wyjścia są stochastyczne $P(x_t|y_t)$
- Można zapisać rozkład prawdopodobieństwa $|C|$ stanów poprzez $|C|$ liczb
- Innymi słowy: w każdym kroku czasowym wybiera się jeden z $|C|$ stanów, więc można zapisać jedynie $\log|C|$ bitów informacji
- Załóżmy, że chce się wykorzystać HMM do modelowania języka tj. przewidywać kolejne słowa. Co trzeba wziąć pod uwagę?
  - semantykę
  - gramatykę
  - ...
- Ile to bitów? 100? Potrzeba więc $2^{100}$ stanów!
- Każdy poznany model ma składniki zależące tylko od jednego (dwóch) poprzednich tagów...

[1] za slajdami Geoffrey Hinton *Neural Networks for Machine Learning*

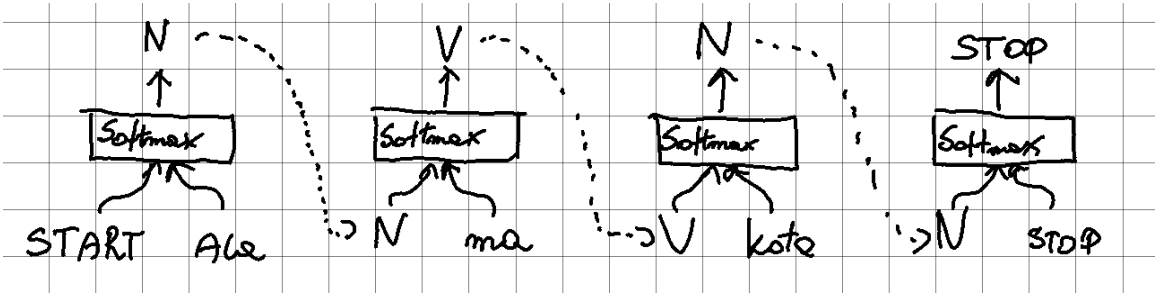
# Ograniczenia Ukrytych Modeli Markova[1]

- HMM przyjmują jeden z $|C|$ stanów ukrytych (np. po jednym dla każdej części mowy), które zmieniają się zgodnie z prawdopodobieństwami tranzycji $P(y_t|y_{t-1})$ a wyjścia są stochastyczne $P(x_t|y_t)$
- Można zapisać rozkład prawdopodobieństwa $|C|$ stanów poprzez $|C|$ liczb
- Innymi słowy: w każdym kroku czasowym wybiera się jeden z $|C|$ stanów, więc można zapisać jedynie $\log|C|$ bitów informacji
- Załóżmy, że chce się wykorzystać HMM do modelowania języka tj. przewidywać kolejne słowa. Co trzeba wziąć pod uwagę?
  - semantykę
  - gramatykę
  - ...
- Ile to bitów? 100? Potrzeba więc $2^{100}$ stanów!
- Każdy poznany model ma składniki zależące tylko od jednego (dwóch) poprzednich tagów...

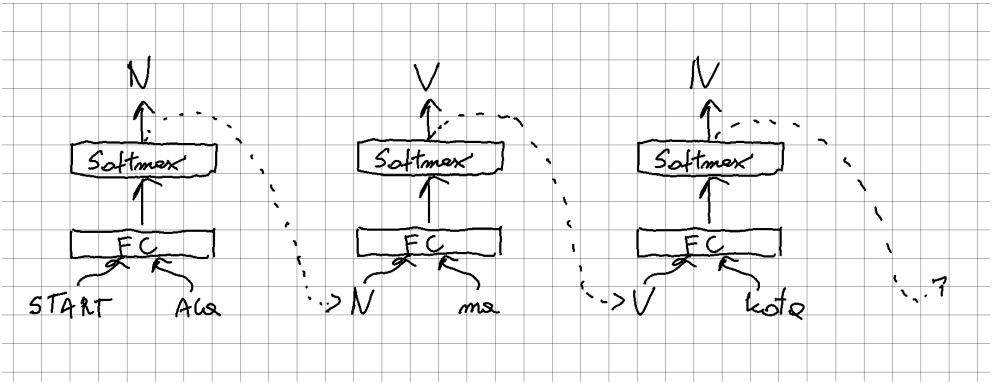
[1] za slajdami Geoffrey Hinton *Neural Networks for Machine Learning*

# MEMM - powtórka

$$P(y_1^n|x_1^n) = \prod_{i=1}^{n} P(y_i|y_{i-1}, x_1^n) = \prod_{i=1}^{n} \frac{e^{\mathbf{w}^T \phi(\mathbf{x}_i, y_{i-1}, y_i)}}{\sum_{y'} e^{\mathbf{w}^T \phi(\mathbf{x}_i, y_{i-1}, y')}}$$

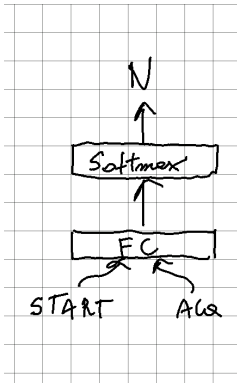


# „Neuronowy” MEMM: Sieć rekurencyjna Jordana




- zastępujemy w MEMM klasyfikator softmax siecią neuronową, otrzymując model rekurencyjny
- nadal ta sieć jest w zasadzie „zwykłą” siecią neuronową, uczoną na specjalnie stworzonym zbiorze uczącym (analogicznie jak dla MEMM)

# Sieć rekurencyjna Jordana




- Warstwa ukryta obliczana wzorem:

$$h_t = \sigma(W[\boxed{\text{START}}; Ala] + b)$$

gdzie zapis $[\boxed{\text{START}}; Ala]$ oznacza konkatenację reprezentacji cech „ $\boxed{\text{START}}$ ” i „Ala”

- Uogólniając:

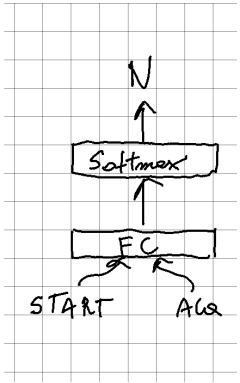
$$h_t = \sigma(W_h[y_{t-1}; x_t] + b_h)$$

$$\hat{y}_t = softmax(W_y h_t + b_y)$$

- Alternatywnie, zamiast konkatenować wejście, można rozbić macierz wag na dwie części:

$$h_t = \sigma(W_{h,y} y_{t-1} + W_{h,x} x_t + b_h)$$

# Sieć rekurencyjna Jordana




- Warstwa ukryta obliczana wzorem:

$$h_t = \sigma(W[\boxed{\text{START}}; Ala] + b)$$

gdzie zapis $[\boxed{\text{START}}; Ala]$ oznacza konkatenację reprezentacji cech „ $\boxed{\text{START}}$ ” i „Ala”

- Uogólniając:

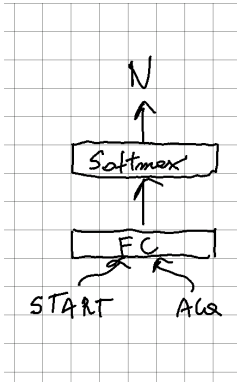
$$h_t = \sigma(W_h[y_{t-1}; x_t] + b_h)$$

$$\hat{y}_t = softmax(W_y h_t + b_y)$$

- Alternatywnie, zamiast konkatenować wejście, można rozbić macierz wag na dwie części:

$$h_t = \sigma(W_{h,y} y_{t-1} + W_{h,x} x_t + b_h)$$

# Sieć rekurencyjna Jordana




- Warstwa ukryta obliczana wzorem:

$$h_t = \sigma(W[\boxed{\text{START}}; Ala] + b)$$

gdzie zapis $[\boxed{\text{START}}; Ala]$ oznacza konkatenację reprezentacji cech „ $\boxed{\text{START}}$ ” i „Ala”

- Uogólniając:

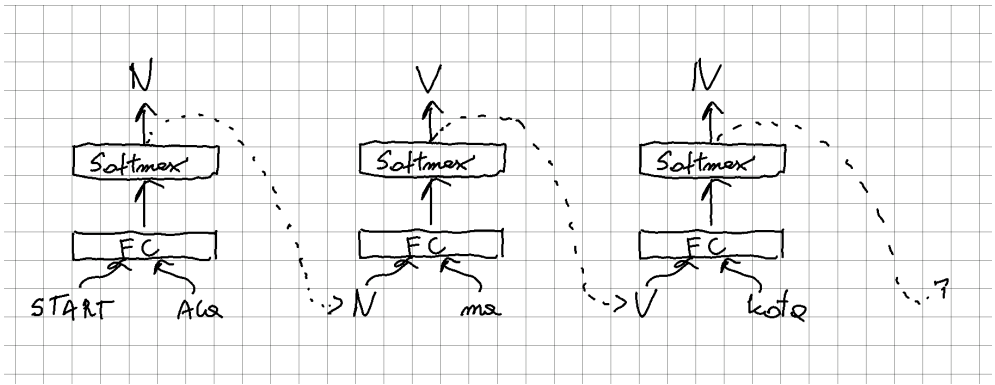
$$h_t = \sigma(W_h[y_{t-1}; x_t] + b_h)$$

$$\hat{y}_t = softmax(W_y h_t + b_y)$$

- Alternatywnie, zamiast konkatenować wejście, można rozbić macierz wag na dwie części:

$$h_t = \sigma(W_{h,y} y_{t-1} + W_{h,x} x_t + b_h)$$

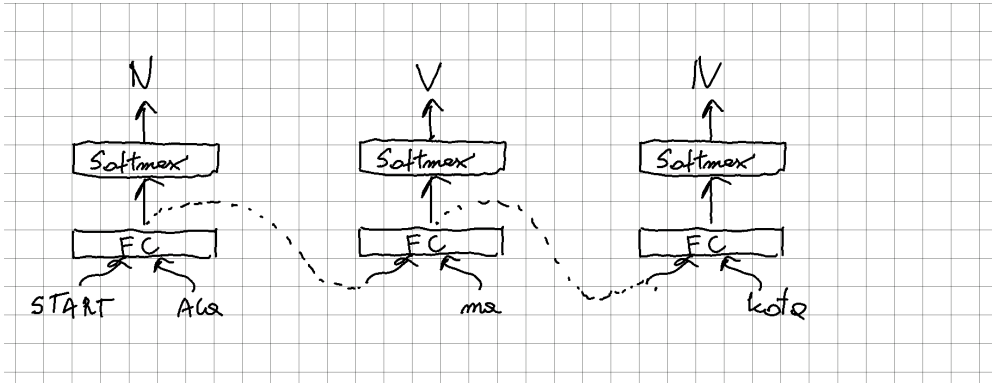
# Sieć rekurencyjna Jordana




$$h_t = \sigma(W_h[y_{t-1}; x_t] + b_h)$$

- Nadal decyzja jest uzależniona tylko od poprzedniej decyzji (choć ta zależy od jeszcze poprzedniej itd.)
- Na wejściu można zastosować warstwę zanurzeń, poprawiając reprezentację $x_t$

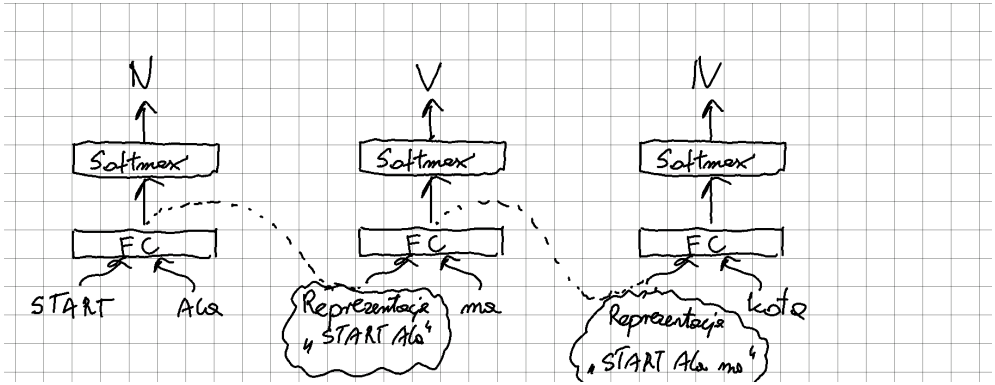
# Sieć rekurencyjna Elmana




$$h_t = \sigma(W_h[h_{t-1}; x_t] + b_h)$$

- Decyzja jest uzależniona tylko od poprzedniej *reprezentacji*

# Sieć rekurencyjna Elmana




$$h_t = \sigma(W_h[h_{t-1}; x_t] + b_h)$$

- Decyzja jest uzależniona tylko od poprzedniej *reprezentacji*
- Ale poprzednia *reprezentacja*, zależy od jeszcze poprzedniej
- RNN: te same wagi w czasie, CNN: te same wagi w przestrzeni

# Sieć rekurencyjna Elmana

Zauważ, że prawdopodobieństwo $P(y_1, y_2, ...y_n|x_1^n)$ można rozbić (bezstratnie) regułą łańcuchową na:

$$P(y_1, y_2, ...y_n|x_1^n) = \prod_{t=1}^{n} P(y_t|x_1^n, y_1^{t-1})$$

W MEMM wprowadziliśmy założenie:

$$P(y_t|x_1^n, y_1^{t-1}) \approx P(y_t|x_1^n, y_{t-1})$$

W modelu rekurencyjnym Elmana wprowadziliśmy założenie:

$$P(y_t|x_1^n, y_1^{t-1}) \approx P(y_t|\underbrace{f(x_1^t, y_1^{t-1})}_{h_t})$$

gdzie $h_t = f(x_1^t) = g(x_t, f(x_1^{t-1})) = g(x_t, g(x_{t-1}, f(x_1^{t-2}))) = \ldots$ jest stanem ukrytym sieci neuronowej będącym wynikiem funkcji, której argumentem jest cała sekwencja $x_1^t$ (i pośrednio lub bezpośrednio $y_1^{t-1}$)

# Sieć rekurencyjna Elmana

Zauważ, że prawdopodobieństwo $P(y_1, y_2, ...y_n|x_1^n)$ można rozbić (bezstratnie) regułą łańcuchową na:

$$P(y_1, y_2, ...y_n|x_1^n) = \prod_{t=1}^{n} P(y_t|x_1^n, y_1^{t-1})$$

W MEMM wprowadziliśmy założenie:

$$P(y_t|x_1^n, y_1^{t-1}) \approx P(y_t|x_1^n, y_{t-1})$$

W modelu rekurencyjnym Elmana wprowadziliśmy założenie:

$$P(y_t|x_1^n, y_1^{t-1}) \approx P(y_t|\underbrace{f(x_1^t, y_1^{t-1})}_{h_t})$$

gdzie $h_t = f(x_1^t) = g(x_t, f(x_1^{t-1})) = g(x_t, g(x_{t-1}, f(x_1^{t-2}))) = \ldots$ jest stanem ukrytym sieci neuronowej będącym wynikiem funkcji, której argumentem jest cała sekwencja $x_1^t$ (i pośrednio lub bezpośrednio $y_1^{t-1}$)

# Sieć rekurencyjna Elmana

Zauważ, że prawdopodobieństwo $P(y_1, y_2, ...y_n|x_1^n)$ można rozbić (bezstratnie) regułą łańcuchową na:

$$P(y_1, y_2, ...y_n|x_1^n) = \prod_{t=1}^{n} P(y_t|x_1^n, y_1^{t-1})$$

W MEMM wprowadziliśmy założenie:

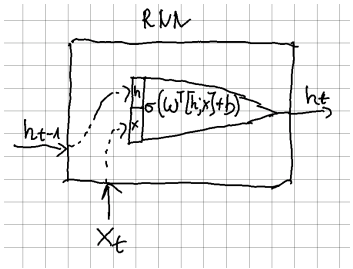
$$P(y_t|x_1^n, y_1^{t-1}) \approx P(y_t|x_1^n, y_{t-1})$$

W modelu rekurencyjnym Elmana wprowadziliśmy założenie:

$$P(y_t|x_1^n, y_1^{t-1}) \approx P(y_t|\underbrace{f(x_1^t, y_1^{t-1})}_{h_t})$$

gdzie $h_t = f(x_1^t) = g(x_t, f(x_1^{t-1})) = g(x_t, g(x_{t-1}, f(x_1^{t-2}))) = \ldots$ jest stanem ukrytym sieci neuronowej będącym wynikiem funkcji, której argumentem jest cała sekwencja $x_1^t$ (i pośrednio lub bezpośrednio $y_1^{t-1}$)

# Neuron rekurencyjny

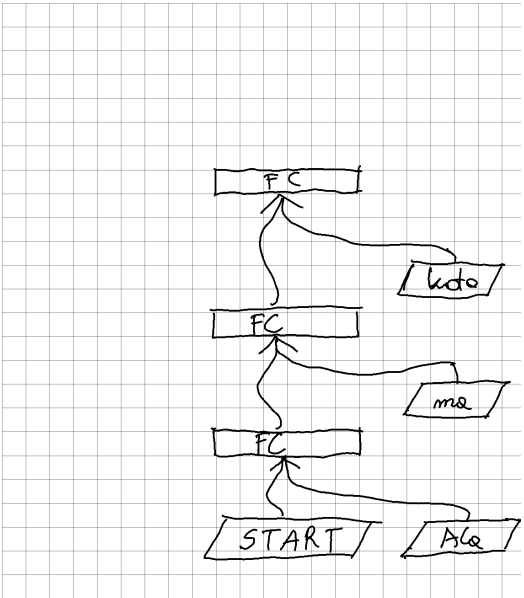



- Neuron rekurencyjny oblicza funkcję:

$$h_t = \sigma(W_h[h_{t-1}; x_t] + b_h)$$

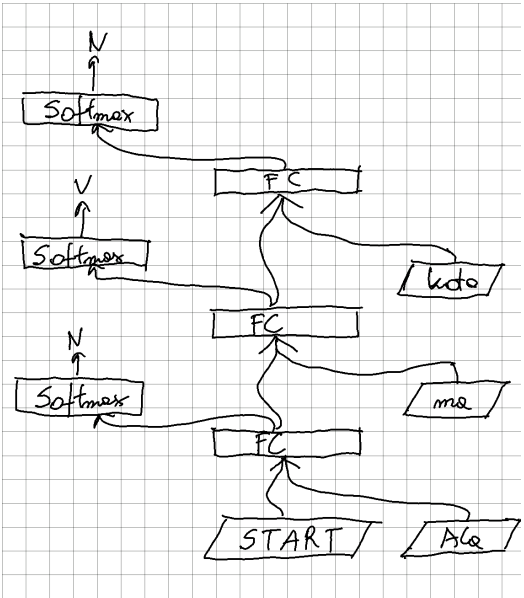
  gdzie $h_t$ to wejście do kolejnych warstw, jak i reprezentacja stanu dla kolejnej chwili czasowej
- $W_h$ nie zmienia się w czasie!
- Uniwersalność: może nauczyć się dowolnej funkcji wykonywalnej na maszynie Turinga.
- Problem wyboru $h_0$ (a.k.a. START )
  - stała wartość $h_0 = 0$
  - stałe zanurzenie, które jest uczone $h_0$

# Sieć rekurencyjna jako sieć w pełni połączona



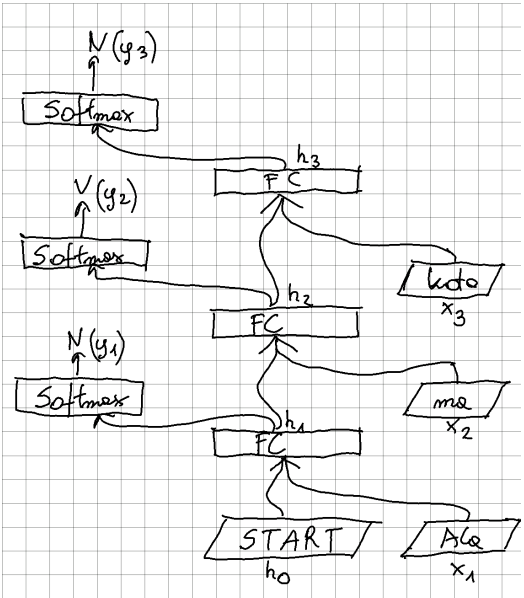

- Sieć ma tyle warstw ile jest chwil czasowych (słów)
- Uogólnianie wiedzy następuje, bo warstwa "FC" ma zawsze te same wagi
- Informacja ucząca (gradient) jest dostarczana poprzez przyłączenie po każdej warstwie warstwy softmax (znów: takie same wagi, niezależnie od $t$)
- Typowy problem sieci wielowarstwowych?

# Sieć rekurencyjna jako sieć w pełni połączona



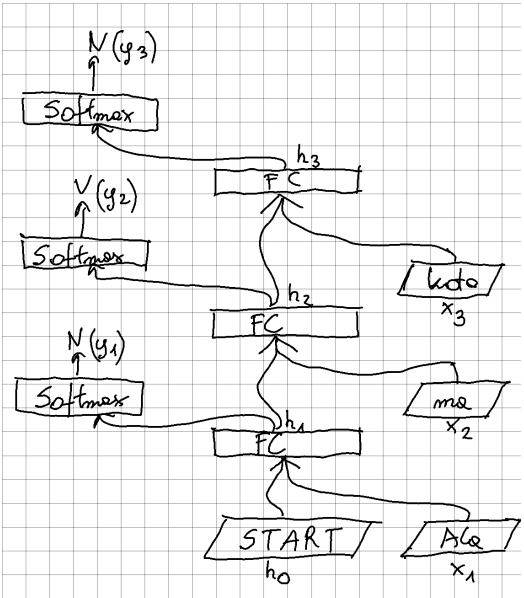

- Sieć ma tyle warstw ile jest chwil czasowych (słów)
- Uogólnianie wiedzy następuje, bo warstwa "FC" ma zawsze te same wagi
- Informacja ucząca (gradient) jest dostarczana poprzez przyłączenie po każdej warstwie warstwy softmax (znów: takie same wagi, niezależnie od $t$)
- Typowy problem sieci wielowarstwowych?

# Sieć rekurencyjna jako sieć w pełni połączona




- Sieć ma tyle warstw ile jest chwil czasowych (słów)
- Uogólnianie wiedzy następuje, bo warstwa "FC" ma zawsze te same wagi
- Informacja ucząca (gradient) jest dostarczana poprzez przyłączenie po każdej warstwie warstwy softmax (znów: takie same wagi, niezależnie od $t$)
- Typowy problem sieci wielowarstwowych?

# Sieć rekurencyjna jako sieć w pełni połączona




- Sieć ma tyle warstw ile jest chwil czasowych (słów)
- Uogólnianie wiedzy następuje, bo warstwa "FC" ma zawsze te same wagi
- Informacja ucząca (gradient) jest dostarczana poprzez przyłączenie po każdej warstwie warstwy softmax (znów: takie same wagi, niezależnie od $t$)
- Typowy problem sieci wielowarstwowych? Zanikający gradient...
  Okazuje się, że w sytuacji współdzielenia macierzy wag ten problem jest jeszcze gorszy...

# Problem eksplodującego/zanikającego gradientu

$$h_t = \sigma(W_{hh} h_{t-1} + W_{hx} x_t + b_h)$$

- Błąd jest liczony po wszystkich wyjściach sieci:

$$L = \sum_{t=1}^{N} \ell(\hat{y}_t, y_t) = \sum_{t=1}^{N} \ell_t$$

- W związku z tym gradient po wagach $W_{hh}$ to

$$\frac{\partial L}{\partial W_{hh}} = \sum_{t=1}^{N} \frac{\partial \ell_t}{\partial W_{hh}}$$

# Problem eksplodującego/zanikającego gradientu[2]

$$h_t = \sigma(\underbrace{W_{hh}h_{t-1} + W_{hx}x_t + b_h}_{z_t})$$

$$\frac{\partial \ell_t}{\partial W_{hh}} = \frac{\partial \ell_t}{\partial h_t}\frac{\partial h_t}{\partial W_{hh}}$$

$$\frac{\partial h_t}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial z_t}{\partial W_{hh}}$$

$$= \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial (W_{hh}h_{t-1} + W_{hx}x_t + b_h)}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial W_{hh}h_{t-1}}{\partial W_{hh}}$$

$$= \frac{\partial \sigma(z_t)}{\partial z_t}\left(h_{t-1}\frac{\partial W_{hh}}{\partial W_{hh}} + W_{hh}\frac{\partial h_{t-1}}{\partial W_{hh}}\right)$$

[2]Wyprowadzenie gradientu jest *błędne* i służy tylko pokazaniu idei (pomija się fakt, że operujemy na macierzach a nie liczbach). Pełne wyprowadzenie w literaturze.

# Problem eksplodującego/zanikającego gradientu[2]

$$h_t = \sigma(\underbrace{W_{hh}h_{t-1} + W_{hx}x_t + b_h}_{z_t})$$

$$\frac{\partial \ell_t}{\partial W_{hh}} = \frac{\partial \ell_t}{\partial h_t}\frac{\partial h_t}{\partial W_{hh}}$$

$$\frac{\partial h_t}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial z_t}{\partial W_{hh}}$$

$$= \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial (W_{hh}h_{t-1} + W_{hx}x_t + b_h)}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial W_{hh}h_{t-1}}{\partial W_{hh}}$$

$$= \frac{\partial \sigma(z_t)}{\partial z_t}\left(h_{t-1}\frac{\partial W_{hh}}{\partial W_{hh}} + W_{hh}\frac{\partial h_{t-1}}{\partial W_{hh}}\right)$$

[2]Wyprowadzenie gradientu jest *błędne* i służy tylko pokazaniu idei (pomija się fakt, że operujemy na macierzach a nie liczbach). Pełne wyprowadzenie w literaturze.

# Problem eksplodującego/zanikającego gradientu[2]

$$h_t = \sigma(\underbrace{W_{hh}h_{t-1} + W_{hx}x_t + b_h}_{z_t})$$

$$\frac{\partial \ell_t}{\partial W_{hh}} = \frac{\partial \ell_t}{\partial h_t}\frac{\partial h_t}{\partial W_{hh}}$$

$$\frac{\partial h_t}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial z_t}{\partial W_{hh}}$$

$$= \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial (W_{hh}h_{t-1} + W_{hx}x_t + b_h)}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial W_{hh}h_{t-1}}{\partial W_{hh}}$$

$$= \frac{\partial \sigma(z_t)}{\partial z_t}\left(h_{t-1}\frac{\partial W_{hh}}{\partial W_{hh}} + W_{hh}\frac{\partial h_{t-1}}{\partial W_{hh}}\right)$$

[2]Wyprowadzenie gradientu jest *błędne* i służy tylko pokazaniu idei (pomija się fakt, że operujemy na macierzach a nie liczbach). Pełne wyprowadzenie w literaturze.

# Problem eksplodującego/zanikającego gradientu[2]

$$h_t = \sigma(\underbrace{W_{hh}h_{t-1} + W_{hx}x_t + b_h}_{z_t})$$

$$\frac{\partial \ell_t}{\partial W_{hh}} = \frac{\partial \ell_t}{\partial h_t}\frac{\partial h_t}{\partial W_{hh}}$$

$$\begin{aligned}\frac{\partial h_t}{\partial W_{hh}} &= \frac{\partial \sigma(z_t)}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial z_t}{\partial W_{hh}} \\ &= \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial (W_{hh}h_{t-1} + W_{hx}x_t + b_h)}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial W_{hh}h_{t-1}}{\partial W_{hh}} \\ &= \frac{\partial \sigma(z_t)}{\partial z_t}\left(h_{t-1}\frac{\partial W_{hh}}{\partial W_{hh}} + W_{hh}\frac{\partial h_{t-1}}{\partial W_{hh}}\right)\end{aligned}$$

[2]Wyprowadzenie gradientu jest *błędne* i służy tylko pokazaniu idei (pomija się fakt, że operujemy na macierzach a nie liczbach). Pełne wyprowadzenie w literaturze.

# Problem eksplodującego/zanikającego gradientu[2]

$$h_t = \sigma(\underbrace{W_{hh}h_{t-1} + W_{hx}x_t + b_h}_{z_t})$$

$$\frac{\partial \ell_t}{\partial W_{hh}} = \frac{\partial \ell_t}{\partial h_t}\frac{\partial h_t}{\partial W_{hh}}$$

$$\begin{aligned}
\frac{\partial h_t}{\partial W_{hh}} &= \frac{\partial \sigma(z_t)}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial z_t}{\partial W_{hh}} \\
&= \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial (W_{hh}h_{t-1} + W_{hx}x_t + b_h)}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial W_{hh}h_{t-1}}{\partial W_{hh}} \\
&= \frac{\partial \sigma(z_t)}{\partial z_t}\left(h_{t-1}\frac{\partial W_{hh}}{\partial W_{hh}} + W_{hh}\frac{\partial h_{t-1}}{\partial W_{hh}}\right)
\end{aligned}$$

[2]Wyprowadzenie gradientu jest *błędne* i służy tylko pokazaniu idei (pomija się fakt, że operujemy na macierzach a nie liczbach). Pełne wyprowadzenie w literaturze.

# Problem eksplodującego/zanikającego gradientu[2]

$$h_t = \sigma(\underbrace{W_{hh}h_{t-1} + W_{hx}x_t + b_h}_{z_t})$$

$$\frac{\partial \ell_t}{\partial W_{hh}} = \frac{\partial \ell_t}{\partial h_t}\frac{\partial h_t}{\partial W_{hh}}$$

$$\begin{aligned}
\frac{\partial h_t}{\partial W_{hh}} &= \frac{\partial \sigma(z_t)}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial z_t}{\partial W_{hh}} \\
&= \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial (W_{hh}h_{t-1} + W_{hx}x_t + b_h)}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial W_{hh}h_{t-1}}{\partial W_{hh}} \\
&= \frac{\partial \sigma(z_t)}{\partial z_t}\left(h_{t-1}\frac{\partial W_{hh}}{\partial W_{hh}} + W_{hh}\frac{\partial h_{t-1}}{\partial W_{hh}}\right)
\end{aligned}$$

[2]Wyprowadzenie gradientu jest *błędne* i służy tylko pokazaniu idei (pomija się fakt, że operujemy na macierzach a nie liczbach). Pełne wyprowadzenie w literaturze.

# Problem eksplodującego/zanikającego gradientu[3]

$$\frac{\partial h_t}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial z_t}\left(h_{t-1} + W_{hh}\frac{\partial h_{t-1}}{\partial W_{hh}}\right)$$

$$= \frac{\partial \sigma(z_t)}{\partial z_t}\left(h_{t-1} + W_{hh}\underbrace{\left(\frac{\partial \sigma(z_{t-1})}{\partial z_{t-1}}\left(h_{t-2} + W_{hh}\frac{\partial h_{t-2}}{\partial W_{hh}}\right)\right)}_{\text{jeszcze raz ten sam wzór}}\right)$$

$$= \frac{\partial \sigma(z_t)}{\partial z_t}h_{t-1} + \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial \sigma(z_{t-1})}{\partial z_{t-1}}W_{hh}h_{t-2} + \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial \sigma(z_{t-1})}{\partial z_{t-1}}W_{hh}W_{hh}\frac{\partial h_{t-2}}{\partial W_{hh}}$$

$$= \sum_{i=1}^{t}\frac{\partial \sigma(z_t)}{\partial z_t}\cdots\frac{\partial \sigma(z_i)}{\partial z_i}W_{hh}^{t-i}h_{i-1}$$

[3]Wyprowadzenie gradientu jest *błędne* i służy tylko pokazaniu idei (pomija się fakt, że operujemy na macierzach, a nie liczbach). Pełne wyprowadzenie w literaturze.

# Problem eksplodującego/zanikającego gradientu[3]

$$\frac{\partial h_t}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial z_t}\left(h_{t-1} + W_{hh}\frac{\partial h_{t-1}}{\partial W_{hh}}\right)$$

$$= \frac{\partial \sigma(z_t)}{\partial z_t}\left(h_{t-1} + W_{hh}\underbrace{\left(\frac{\partial \sigma(z_{t-1})}{\partial z_{t-1}}\left(h_{t-2} + W_{hh}\frac{\partial h_{t-2}}{\partial W_{hh}}\right)\right)}_{\text{jeszcze raz ten sam wzór}}\right)$$

$$= \frac{\partial \sigma(z_t)}{\partial z_t}h_{t-1} + \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial \sigma(z_{t-1})}{\partial z_{t-1}}W_{hh}h_{t-2} + \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial \sigma(z_{t-1})}{\partial z_{t-1}}W_{hh}W_{hh}\frac{\partial h_{t-2}}{\partial W_{hh}}$$

$$= \sum_{i=1}^{t}\frac{\partial \sigma(z_t)}{\partial z_t}\cdots\frac{\partial \sigma(z_i)}{\partial z_i}W_{hh}^{t-i}h_{i-1}$$

[3]Wyprowadzenie gradientu jest *błędne* i służy tylko pokazaniu idei (pomija się fakt, że operujemy na macierzach, a nie liczbach). Pełne wyprowadzenie w literaturze.

# Problem eksplodującego/zanikającego gradientu[3]

$$\frac{\partial h_t}{\partial W_{hh}} = \frac{\partial \sigma(z_t)}{\partial z_t}\left(h_{t-1} + W_{hh}\frac{\partial h_{t-1}}{\partial W_{hh}}\right)$$

$$= \frac{\partial \sigma(z_t)}{\partial z_t}\left(h_{t-1} + W_{hh}\underbrace{\left(\frac{\partial \sigma(z_{t-1})}{\partial z_{t-1}}\left(h_{t-2} + W_{hh}\frac{\partial h_{t-2}}{\partial W_{hh}}\right)\right)}_{\text{jeszcze raz ten sam wzór}}\right)$$

$$= \frac{\partial \sigma(z_t)}{\partial z_t}h_{t-1} + \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial \sigma(z_{t-1})}{\partial z_{t-1}}W_{hh}h_{t-2} + \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial \sigma(z_{t-1})}{\partial z_{t-1}}W_{hh}W_{hh}\frac{\partial h_{t-2}}{\partial W_{hh}}$$

$$= \sum_{i=1}^{t}\frac{\partial \sigma(z_t)}{\partial z_t}\cdots\frac{\partial \sigma(z_i)}{\partial z_i}W_{hh}^{t-i}h_{i-1}$$

[3]Wyprowadzenie gradientu jest *błędne* i służy tylko pokazaniu idei (pomija się fakt, że operujemy na macierzach, a nie liczbach). Pełne wyprowadzenie w literaturze.

# Problem eksplodującego/zanikającego gradientu[3]

$$
\begin{aligned}
\frac{\partial h_t}{\partial W_{hh}} &= \frac{\partial \sigma(z_t)}{\partial z_t}\left(h_{t-1} + W_{hh}\frac{\partial h_{t-1}}{\partial W_{hh}}\right) \\
&= \frac{\partial \sigma(z_t)}{\partial z_t}\left(h_{t-1} + W_{hh}\underbrace{\left(\frac{\partial \sigma(z_{t-1})}{\partial z_{t-1}}\left(h_{t-2} + W_{hh}\frac{\partial h_{t-2}}{\partial W_{hh}}\right)\right)}_{\text{jeszcze raz ten sam wzór}}\right) \\
&= \frac{\partial \sigma(z_t)}{\partial z_t}h_{t-1} + \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial \sigma(z_{t-1})}{\partial z_{t-1}}W_{hh}h_{t-2} + \frac{\partial \sigma(z_t)}{\partial z_t}\frac{\partial \sigma(z_{t-1})}{\partial z_{t-1}}W_{hh}W_{hh}\frac{\partial h_{t-2}}{\partial W_{hh}} \\
&= \sum_{i=1}^{t}\frac{\partial \sigma(z_t)}{\partial z_t}\cdots\frac{\partial \sigma(z_i)}{\partial z_i}W_{hh}^{t-i}h_{i-1}
\end{aligned}
$$

[3]Wyprowadzenie gradientu jest *błędne* i służy tylko pokazaniu idei (pomija się fakt, że operujemy na macierzach, a nie liczbach). Pełne wyprowadzenie w literaturze.

# Problem eksplodującego/zanikającego gradientu

- Gradient (a więc i jego norma) jest uzależniony od potęgi $W_{hh}^t$
- Rozważając wielkość gradientu powiązanego z elementem sumy do najdalszego stanu ukrytego (początek sekwencji)

$$\left\| \frac{\partial \sigma(z_t)}{\partial z_t} \cdots \frac{\partial \sigma(z_1)}{\partial z_1} W_{hh}^{t-1} h_0 \right\| \leq C ||W_{hh}||^{t-1}$$

gdzie $C$ jest ograniczeniem górnym pozostałych elementów. Zauważ, że pochodne typowych funkcji aktywacji $\frac{\partial \sigma(z_i)}{\partial z_i} \leq 1$

- Jeżeli $||W_{hh}|| < 1$ - bardzo szybko gradient staje się bardzo mały ($0.5^{10} \approx 0.000977$)

$\Rightarrow$ w praktyce RNN modelują zależności maksymalnie 7-gramowe

- Jeżeli $||W_{hh}|| > 1$ - bardzo szybko gradient staje się bardzo duży ($5^{10} = 9\,765\,625$)

$\Rightarrow$ wagi w modelu wynoszą NaN, całkowite niepowodzenie procesu uczenia się

- Jak to wpływa na możliwość modelowania długich zależności?

# Problem eksplodującego/zanikającego gradientu

- Problem eksplodującego gradientu można stosunkowo prosto rozwiązać, poprzez wprowadzenie górnego ograniczenia na jego wielkość (ang. *gradient clipping*)

## Normalizacja gradientu

1. Oblicz gradient funkcji celu względem wag

$$\Delta = \frac{\partial L}{\partial W}$$

2. Jeżeli $\|\Delta\| \geq \gamma$ to

$$\Delta = \frac{\gamma}{\|\Delta\|}\Delta$$

- Problem zanikającego gradientu, rozwiązać już nie tak łatwo...

# Long short-term memory (LSTM) - intuicja

- Chcielibyśmy rozwiązać problem zanikającego gradientu, który uniemożliwia pamiętanie długich zależności...
- Naturalną koncepcją informatyczną do pamiętania informacji jest ... pamięć (!)
- Pomysł: stwórzmy sieć neuronową, która nauczy się korzystać z komórki pamięci

# Long short-term memory (LSTM) - komórka pamięci

- Zaprojektujmy komórkę pamięci, która będzie przechowywała pewną wartość oraz będzie miała możliwość wykonania dwóch operacji na tej wartości:
  1. wyczyścić ją (wyzerować)
  2. zwiększyć ją/zmniejszyć ją o pewną wartość
- Potrzeba w związku z tym trzech sygnałów sterujących komórką:
  - sygnał „keep" (zanegowany „forget"[4]) – sygnał binarny czyszczący pamięć dla wartości 0
  - sygnał „action" – sygnał binarny, ale dla wygody $a \in \{-1, 1\}$ oznaczający zmniejszanie i zwiększanie
  - sygnał „input" – wartość binarna, o którą będzie zwiększana/zmniejszana pamięć

[4] Sygnał „keep" zwykle oznacza się literką $f$...

# Long short-term memory (LSTM) - komórka pamięci

- Zaprojektujmy komórkę pamięci, która będzie przechowywała pewną wartość oraz będzie miała możliwość wykonania dwóch operacji na tej wartości:
  1. wyczyścić ją (wyzerować)
  2. zwiększyć ją/zmniejszyć ją o pewną wartość
- Potrzeba w związku z tym trzech sygnałów sterujących komórką:
  - sygnał „keep" (zanegowany „forget"[4]) – sygnał binarny czyszczący pamięć dla wartości 0
  - sygnał „action" – sygnał binarny, ale dla wygody $a \in \{-1, 1\}$ oznaczający zmniejszanie i zwiększanie
  - sygnał „input" – wartość binarna, o którą będzie zwiększana/zmniejszana pamięć

[4] Sygnał „keep" zwykle oznacza się literką $f$...

# Long short-term memory (LSTM) - sygnały sterujące

Potrzeba w związku z tym trzech sygnałów tj. neuronów sterujących komórką:

- sygnał „keep" (zanegowany „forget") – sygnał binarny czyszczący pamięć dla wartości 0

$$f_t = \sigma(w_f^T [h_{t-1}; x_t])$$

- sygnał „action" – sygnał binarny, ale dla wygody $a \in \{-1, 1\}$ oznaczający zmniejszanie i zwiększanie

$$a_t = tanh(w_a^T [h_{t-1}; x_t])$$

- sygnał „input" – wartość binarna, o którą będzie zwiększana/zmniejszana pamięć

$$i_t = \sigma(w_i^T [h_{t-1}; x_t])$$

# Long short-term memory (LSTM) - sygnały sterujące

$$f_t = \sigma(w_f^T [h_{t-1}; x_t])$$
$$a_t = tanh(w_a^T [h_{t-1}; x_t])$$
$$i_t = \sigma(w_i^T [h_{t-1}; x_t])$$

A także implementujemy działanie tych sygnałów na zapisanej wartości $c$ w pamięci.

$$c_t = f_t \cdot c_{t-1} + a_t \cdot i_t$$

Ponadto potrzeba, aby neuron (oprócz obsłużenia pamięci) zwrócił też jakąś wartość na wyjście. Wprowadzamy więc też sygnał „output", który kontroluje wynik [5] :

$$o_t = \sigma(w_o^T [h_{t-1}; x_t])$$

$$h_t = o_t \cdot tanh(c_t)$$

[5] Jest to jedna z możliwych implementacji tych intuicji – patrz kurs prof. Krawca „Głębokie uczenie maszynowe" i pracę Greff et al. *LSTM: A Search Space Odyssey*

# Long short-term memory (LSTM) - sygnały sterujące

$$f_t = \sigma(w_f^T [h_{t-1}; x_t])$$
$$a_t = tanh(w_a^T [h_{t-1}; x_t])$$
$$i_t = \sigma(w_i^T [h_{t-1}; x_t])$$

A także implementujemy działanie tych sygnałów na zapisanej wartości $c$ w pamięci.

$$c_t = f_t \cdot c_{t-1} + a_t \cdot i_t$$

Ponadto potrzeba, aby neuron (oprócz obsłużenia pamięci) zwrócił też jakąś wartość na wyjście. Wprowadzamy więc też sygnał „output", który kontroluje wynik [5] :

$$o_t = \sigma(w_o^T [h_{t-1}; x_t])$$

$$h_t = o_t \cdot tanh(c_t)$$

[5] Jest to jedna z możliwych implementacji tych intuicji – patrz kurs prof. Krawca „Głębokie uczenie maszynowe" i pracę Greff et al. *LSTM: A Search Space Odyssey*

# Dalsze problemy RNN...

- Czasami jedna warstwa nie wystarcza by zamodelować użyteczne cechy
  $\Rightarrow$ dodaj więcej warstw rekurencyjnych
- RNN podejmują decyzję jedynie na podstawie początku sekwencji
  $\Rightarrow$ sieci dwukierunkowe
- RNN dzielą wiele podobieństw z MEMM, w tym lokalną normalizację wyjścia i lokalne decyzje (czy nie ma problemu obciążenia etykietą?)
  $\Rightarrow$ sieci dwukierunkowe (?)
  $\Rightarrow$ warstwa CRF

# Dwukierunkowe sieci RNN

- W modelu rekurencyjnym wprowadziliśmy założenie:

$$P(y_t|x_1^n, y_1^{t-1}) \approx P(y_t|\underbrace{f(x_1^t)}_{h_t})$$

  gdzie $h_t = \sigma(W_h[h_{t-1}; x_t] + b_h)$
- Ponieważ wytworzone cechy $h_i$ są jedyną przesłanką do emisji $y_i$ to trochę nadwyrężając[6] można interpretować że $h_t = f(x_1^t, y_1^{t-1})$, a więc do bezstratnej[7] dekompozycji brakuje jedynie modelowania końcówki zdania $x_{t+1}^n$.
- Ponadto predykcje wykonywane są zachłannie, model w ogóle nie rozważa przyszłych elementów sekwencji
- Pomysł: umożliwienie sieci patrzenia zarówno w przód jak i tył zdania.

[6] Można też dokonać prostej modyfikacji, w każdej iteracji dodatkowo konkatenując $y_{t-1}$ do wejścia neuronu, czyniąc tę interpretację w pełni prawidłową

[7] Hiperoptymistycznie zakładając, że $h_t$ zachowuje wszystkie wymagane informacje

# Dwukierunkowe sieci RNN

- Jedna warstwa sieci dwukierunkowej w rzeczywistości posiada dwie warstwy „zwykłych" neuronów rekurencyjnych: patrzące „w przód" i „w tył"

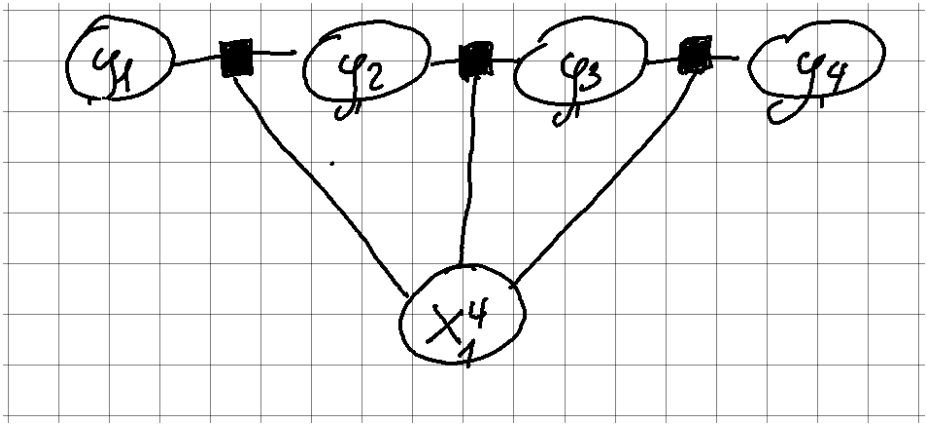
$$h_t^{(tyl)} = \sigma(W_1[h_{t+1}^{(tyl)}; x_t] + b_1)$$

$$h_t^{(przod)} = \sigma(W_2[h_{t-1}^{(przod)}; x_t] + b_2)$$

  Potencjalnie różne wektory początkowe $h_0^{(przod)}$ i $h_n^{(tyl)}$ (analogia do START , STOP ).
- Pierwsze neurony analizują sekwencję od lewej do prawej, drugie od prawej do lewej
- Końcowy wynik warstwy to konkatenacja obydwu wyników:

$$h_t = [h_t^{(przod)}; h_t^{(tyl)}]$$

# Warunkowe pola losowe - powtórzenie




$$P(\mathrm{y}|\mathrm{x}) = \frac{1}{Z(\mathrm{x})} \prod_{i=1}^{n} \Psi_i(y_i, y_{i-1}, \mathrm{x})$$

# Warunkowe pola losowe jako warstwa NN

$$P(\mathrm{y}|\mathrm{x}) = \frac{1}{Z(\mathrm{x})} \prod_{i=1}^{n} \Psi_i(y_i, y_{i-1}, \mathrm{x})$$

- Dotychczas funkcje oceny były liczone wyrażeniem liniowym $\Psi_i(y_i, y_{i-1}, \mathrm{x}) = e^{\mathrm{w}^T \phi(y_i, y_{i-1}, \mathrm{x})}$, a cechy były ręcznie projektowane
- Sieć neuronowa to bardzo dobry ekstraktor cech – wyjścia z sieci rekurencyjnej $h_t$ w kolejnych iteracjach potraktujmy jako wektory cech!

# Warunkowe pola losowe jako warstwa NN

Najczęstsze sposoby implementacji:

- rozbicie funkcji oceny na ocenę emisji i tranzycji

$$\Psi_i(y_i, y_{i-1}, \mathrm{x}) = \exp\left(\psi(y_i, \mathrm{x}) + \psi(y_i, y_{i-1})\right)$$

  Na przykład, ocena stanu przez RNN (emisja) oraz wyraz wolny związany z tranzycją

$$\Psi_i(y_i, y_{i-1}, \mathrm{x}) = \exp\left(\mathrm{W}_{y_i} h_i + \mathrm{b}_{y_i, y_{i-1}}\right)$$

- wykorzystanie RNN do pełnej oceny tranzycji i emisji

$$\Psi_i(y_i, y_{i-1}, \mathrm{x}) = \exp\left(\mathrm{W}_{y_i, y_{i-1}} h_i + \mathrm{b}_{y_i, y_{i-1}}\right)$$

- połączenie powyższych i różne inne modyfikacje...

$$\Psi_i(y_i, y_{i-1}, \mathrm{x}) = \exp\left(\mathrm{W}_{y_i, y_{i-1}} h_i + \mathrm{W}_{y_i} h_i + \mathrm{b}_{y_i}\right)$$

# Warunkowe pola losowe jako warstwa NN

- Wykorzystując sieć z warstwą CRF, cała sieć zamienia się w ekstraktor użytecznych cech dla klasyfikatora CRF (tak jak przy softmax zamienia się w ekstrakcję cech dla klasyfikacji)
- Algorytm uczenia:
  - Dla każdej epoki i paczki danych:
    - Wykonaj propagację w przód sieci neuronowej np. LSTM
    - Obliczając funkcje oceny na podstawie otrzymanych reprezentacji, wykonaj algorytm propagacji przekonania („w przód i w tył") i oblicz prawdopodobieństwo wg. CRF
    - Zaktualizuj wagi CRF (zgodnie z gradientami)
    - Gradienty CRF propaguj dalej przez sieć neuronową (propagacja wsteczna)
- Predykcję, po obliczeniu cech, wykonuje się np. algorytmem Viterbiego

# Przykładowe wyniki sieci rekurencyjnych typu LSTM [8]

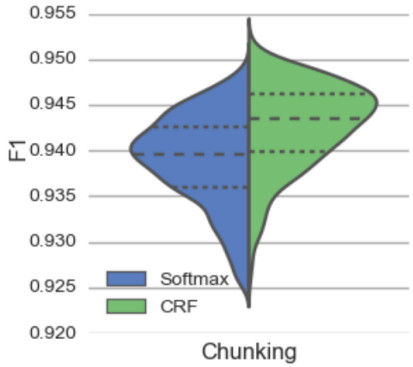
| | POS | | NER | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Dev | | | Test | | |
| Model | Dev Acc. | Test Acc. | Prec. | Recall | F1 | Prec. | Recall | F1 |
| BiRNN | 96.56 | 96.76 | 92.04 | 89.13 | 90.56 | 87.05 | 83.88 | 85.44 |
| BiLSTM | 96.88 | 96.93 | 92.31 | 90.85 | 91.57 | 87.77 | 86.23 | 87.00 |
| BiLSTM-CNN | 97.34 | 97.33 | 92.52 | 93.64 | 93.07 | 88.53 | 90.21 | 89.36 |
| BiLSTM-CNN-CRF | 97.46 | 97.55 | 94.85 | 94.63 | 94.74 | 91.35 | 91.06 | 91.21 |

- Tagowanie częściami mowy (POS): fragment Penn Treebank (Wall Street Journal), 45 tagów
- Wykrywanie encji nazwanych (NER): CoNLL 2003, 4 rodzaje encji, tagowanie BIOES
- Warianty z „CNN" używają 30-wymiarowych zanurzeń znaków, łączonych w reprezentację słów przez CNN (30 filtrów, długość 3, max-pooling over time)
- Warstwa LSTM używa 200 neuronów, zanurzenia słów mają 100 wymiarów, stany początkowe inicjalizowane $h_0 = 0$, normalizacja gradientu z $\gamma = 5$

[8] Ma & Hovy, *End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF*

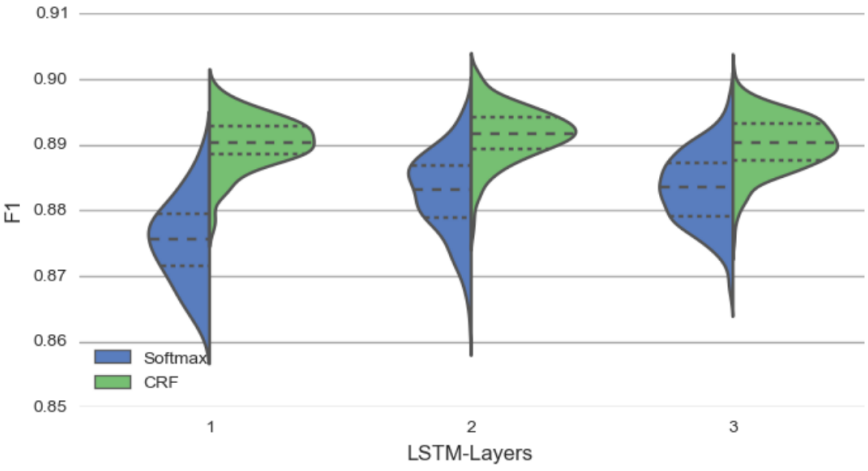
# Warunkowe pola losowe jako warstwa NN

- „Using a CRF instead of a softmax classifieras a last layer gave *a large performance increase* for tasks with a high dependency between tags. This was also true for stacked-BiLSTM layers."[9]




[9] „Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks"

# Warunkowe pola losowe jako warstwa NN – wpływ liczby warstw



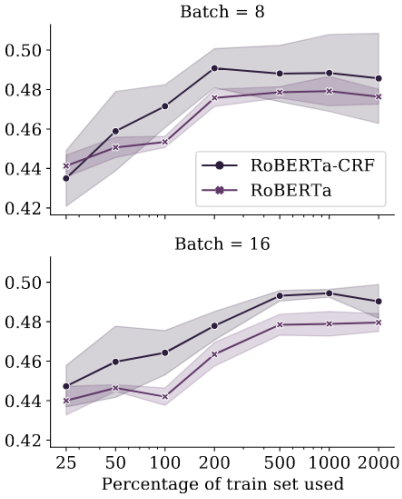

Zadanie: CoNLL 2003 NER, sieć dwukierunkowa LSTM

# Obserwacje z dużego studium eksperymentalnego[10]

| parametr | wpływ na wynik | najlepsza wartość |
|---|---|---|
| zanurzenia słów | duży | specjalizowane, uwzględniające syntaktykę ⇒ kolejny moduł |
| kodowanie znaków | niski | CNN (vs LSTM, bo są szybsze) |
| optymalizator | duży | Adam z momentem Nesterov'a |
| normalizacja/ucinanie gradientu | duży | Normalizacja gradientu |
| schemat tagowania | umiarkowany | BIO |
| klasyfikator | duży | CRF |
| dropout | duży | variational dropout |
| liczba warstw LSTM | umiarkowany | 2 |
| liczba neuronów ukrytych | niski | 100 |

[10] Reimers et al. *Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks*

# Ciekawostka: CRF + Transformers




- Zwycięskie rozwiązanie na konkursie SemEval-2020 Task 11 (wykrywanie propagandy w tekstach)
- Jako ekstraktor cech: sieć typu transformer
- Polski zespół, ApplicaAI ;)

Do zobaczenia!

Fundusze Europejskie
Polska Cyfrowa

Rzeczpospolita Polska

Unia Europejska
Europejski Fundusz Rozwoju Regionalnego