Recurrent neural network

Марина Горлова

Основная идея

Будем использовать вход как последовательную информацию.

x_t - вход на шаге t

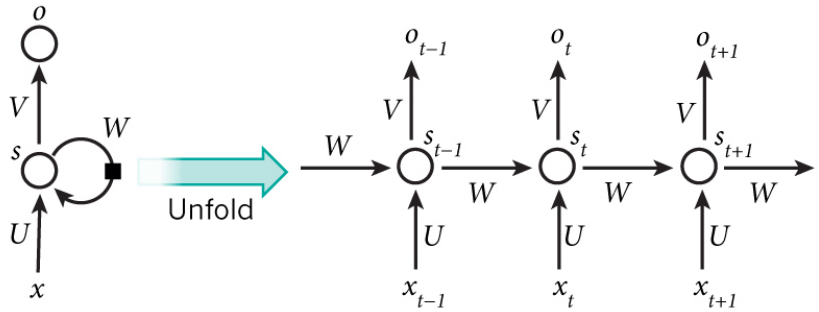
s_t - скрытое состояние на шаге t

$$s_t = f(Ux_t + Ws_{t-1})$$

где f - нелинейная функция tanh, ReLU

o_t - выход на шаге t

например $o_t = \mathrm{softmax}(Vs_t)$



Что важно отметить:

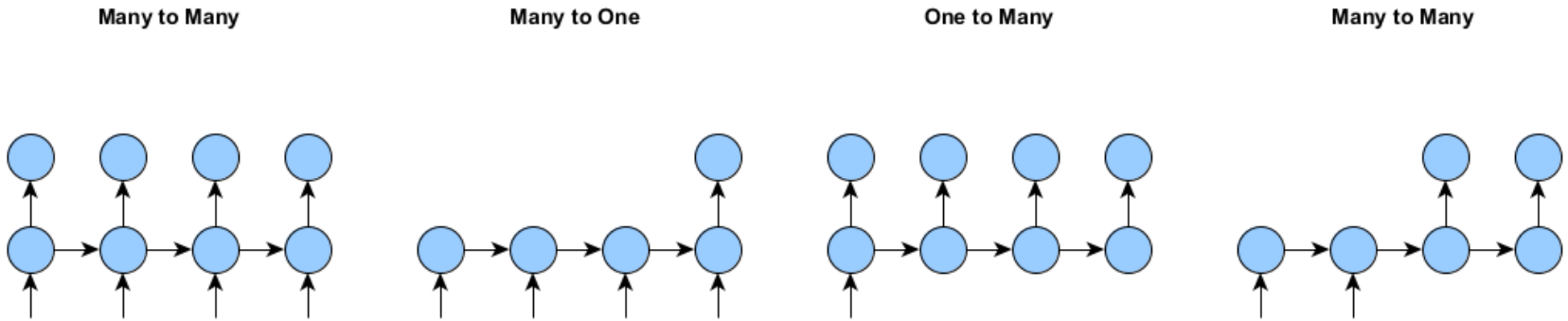
s_t можно интерпретировать как память в момент времени t

o_t вычисляется только на основе s_t

U, V, W - одни и те же параметры, обучаемые во времени t

не обязательно выдавать o_t для каждого x_t

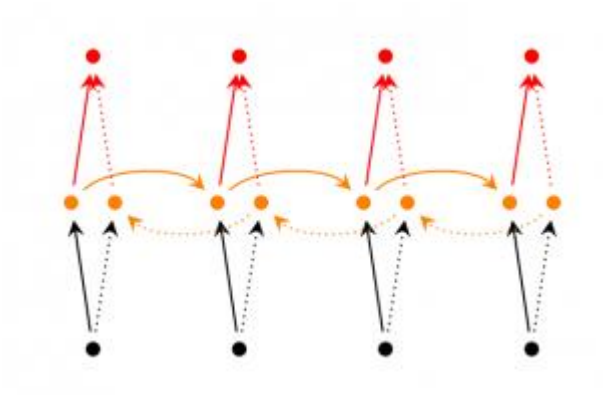
Архитектуры RNN



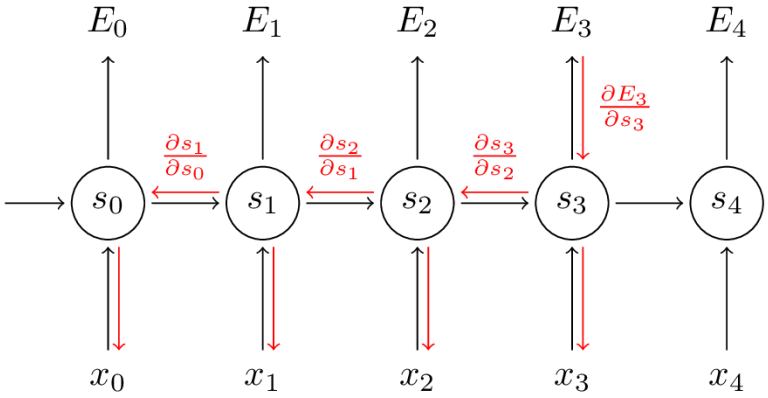
Bidirectional RNN

Важны не только данные до шага t, но и после него.

По сути две сети, выходы которых конкатенируются для каждого шага.



Backpropagation Through Time (BPTT)



$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$

$$\frac{\partial s_3}{\partial s_1} = \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial s_1}$$

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \left(\prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W}$$

Из-за большого количества произведений RNN подвержены проблеме Vanishing / Exploding gradient

Vanishing gradient

Иная инициализация весов, регуляризация, ReLU

Но более популярное решение - использовать

LSTM (Long Short Term Memory) networks

1997

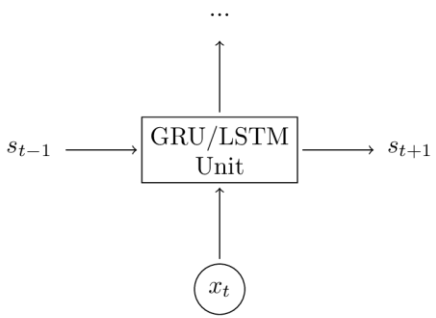
GRUs (Gated Recurrent Units)

2014, облегченная LSTM

LSTM

Справляется с vanishing gradient с помощью механизма gating

Важно помнить, что это всего лишь иной способ вычисления s_t!



LSTM

- обозначает поэлементное умножение

$$
\begin{aligned}
i &= \sigma(x_t U^i + s_{t-1} W^i) \\
f &= \sigma(x_t U^f + s_{t-1} W^f) \\
o &= \sigma(x_t U^o + s_{t-1} W^o) \\
g &= \ tanh(x_t U^g + s_{t-1} W^g) \\
c_t &= c_{t-1} \circ f + g \circ i \\
s_t &= \tanh(c_t) \circ o
\end{aligned}
$$

i, f, o – “gates” принимают значения от 0 до 1

- i – gate сохранения вычисленного состояния
- f – gate забывания вычисленного состояния
- f – gate передачи вычисленного состояния в следующий элемент

g – кандидат в s_t, c_t – внутренняя память, s_t – финальное скрытое состояние

GRU: отличия от LSTM

z – update gate

r – reset gate

$$z = \sigma(x_t U^z + s_{t-1} W^z)$$
$$r = \sigma(x_t U^r + s_{t-1} W^r)$$
$$h = tanh(x_t U^h + (s_{t-1} \circ r) W^h)$$
$$s_t = (1 - z) \circ h + z \circ s_{t-1}$$

- GRU – 2 gates, LSTM – 3 gates
- GRU не вычисляет внутреннюю память
- i и f ~ u, o ~ z и r
- GRU не использует вторую нелинейность для вычисления S_t

Dropout

Рекуррентные слои имеют 2 типа dropout:

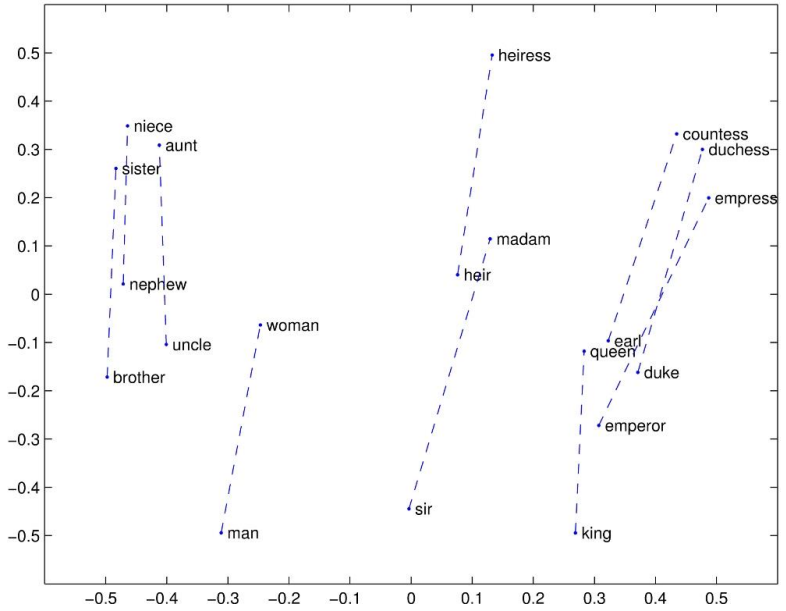
- Dropout: маска применяется ко входному слою
- Recurent_dropout: одинаковая маска применяется для всех recurrent unit

Работа с текстом: embedding

Представление слов вектором размерности меньше объема словаря.

Word2vec, GloVe

anarchism	0.5	0.1	−0.1
originated	−0.5	0.3	0.9
as	0.3	−0.5	−0.3
a	0.7	0.2	−0.3
term	0.8	0.1	−0.1
of	0.4	−0.6	−0.1
abuse	0.7	0.1	−0.4



Keras embedding layer

input_dim – размер словаря, output_dim – размерность представления,

input_length – длинна текса

```
model = Sequential()
model.add(Embedding(input_dim=1000, output_dim=64, input_length=10))
# the model will take as input an integer matrix of size (batch, input_length).
# the largest integer (i.e. word index) in the input should be no larger than 999 (vocabulary size).
# now model.output_shape == (None, 10, 64), where None is the batch dimension.

input_array = np.random.randint(1000, size=(32, 10))

model.compile('rmsprop', 'mse')
output_array = model.predict(input_array)
assert output_array.shape == (32, 10, 64)
```

Word2vec

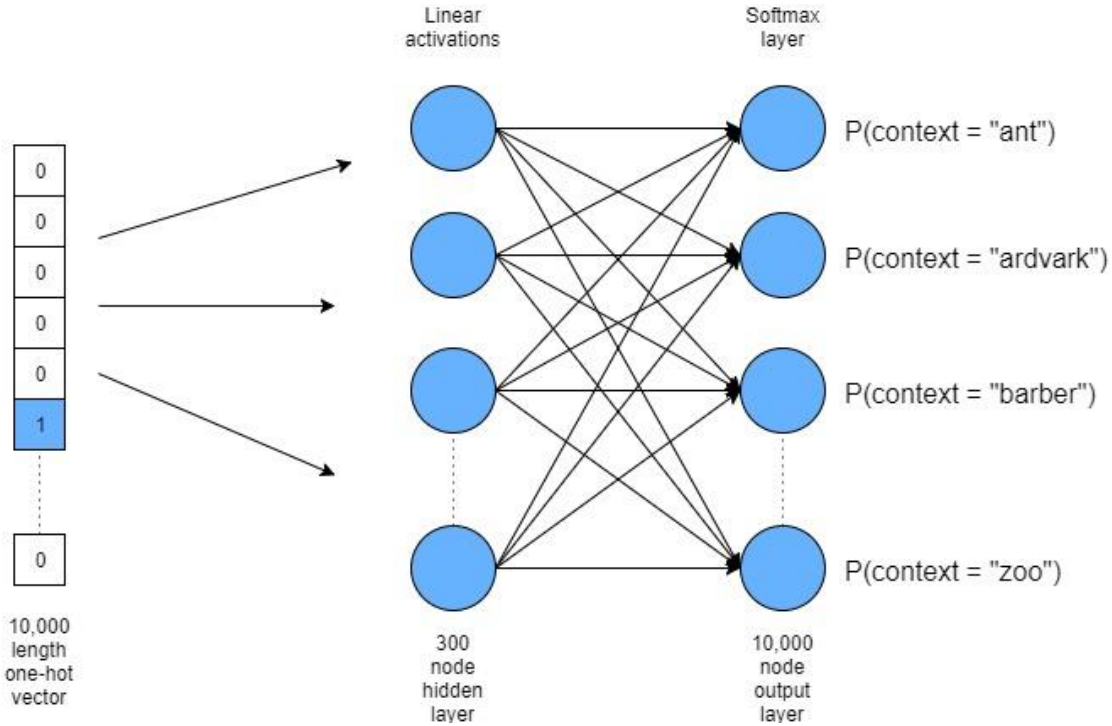
Архитектуры:

- continuous bag-of-words (CBOW) (предсказываем слово из контекста)
- continuous skip-gram (предсказываем контекст по слову)

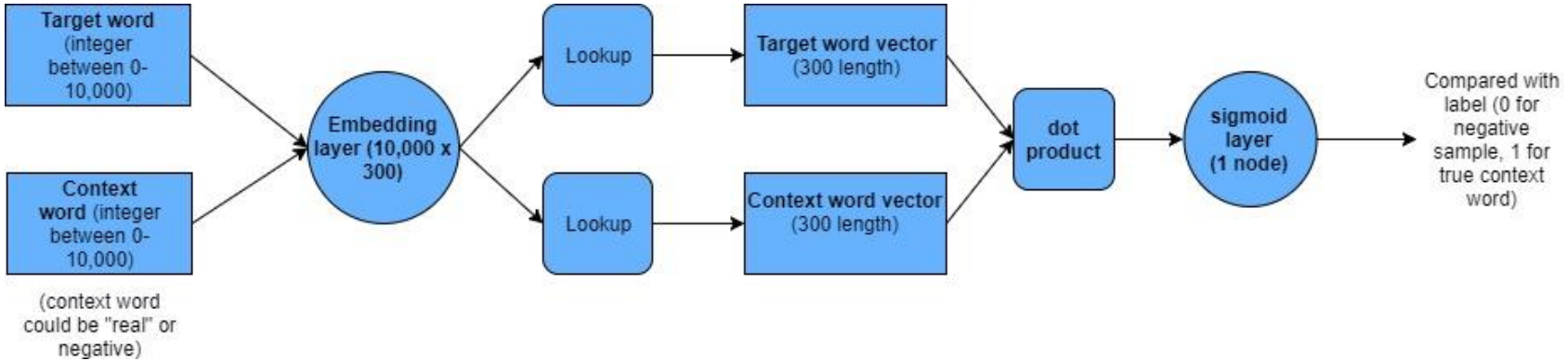
Оптимизация:

- softmax
- negative sampling

Word2vec: softmax



Word2vec: softmax issue and negative sampling



$$similarity = cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2}$$