РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ

Сергей Николенко

СПбГУ — Санкт-Петербург
05 ноября 2020 г.

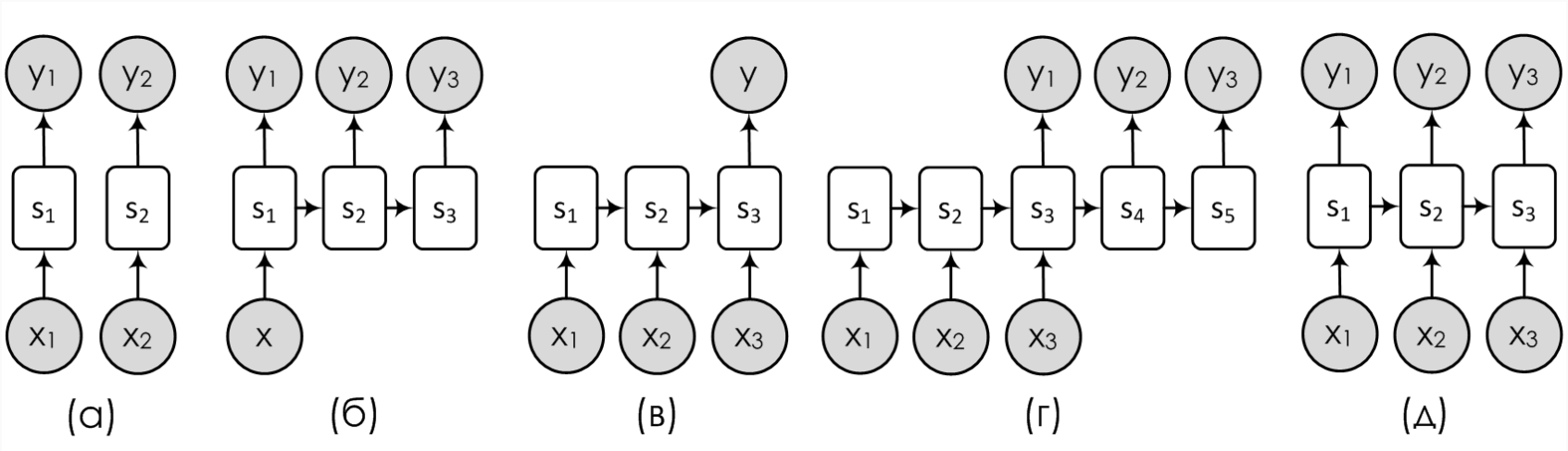
Random facts:

- 5 ноября в Великобритании — Guy Fawkes' Night; в эту ночь отмечается провал Порохового заговора, когда группа католиков-заговорщиков попыталась взорвать Парламент Великобритании в Лондоне в ночь на 5 ноября 1605 года
- 5 ноября 1935 г. регулярное троллейбусное движение открылось в Киеве, 5 ноября 1952 г. — во Владимире, 5 ноября 1964 г. — в Чернигове
- 5 ноября 1898 г. филиппинские националисты на острове Негрос восстали против испанского господства и основали Республику Негрос, которая просуществовала три года и стала протекторатом США
- 5 ноября 1917 г. Тихон был избран патриархом Московским, а Владимир Ленин инициировал Октябрьскую революцию
- 5 ноября 1940 г. Франклин Рузвельт стал первым и единственным президентом США, избранным на третий срок
- 5 ноября 1996 г. президенту России Борису Ельцину сделали операцию шунтирования на сердце; обязанности Президента РФ в это время исполнял Виктор Черномырдин

Рекуррентные нейронные сети

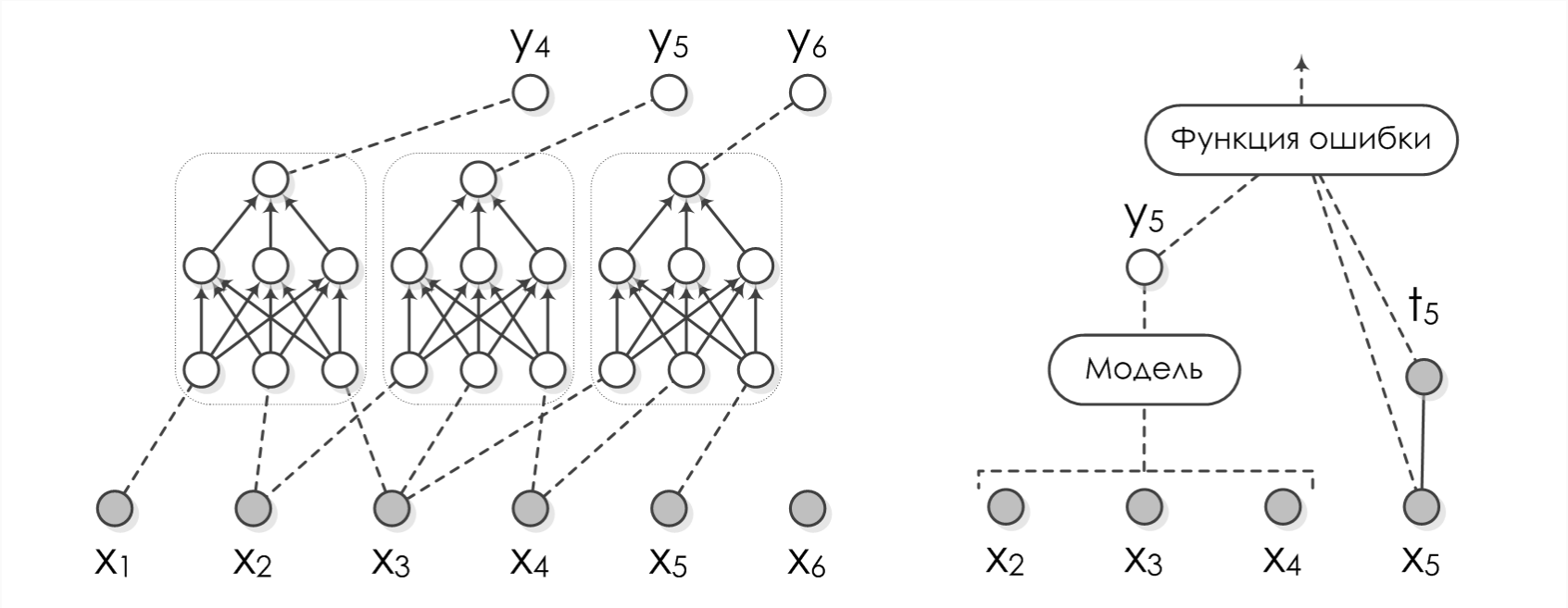
Последовательности

- Последовательности: текст, временные ряды, речь, музыка...
- Есть разные виды задач, основанных на последовательностях:



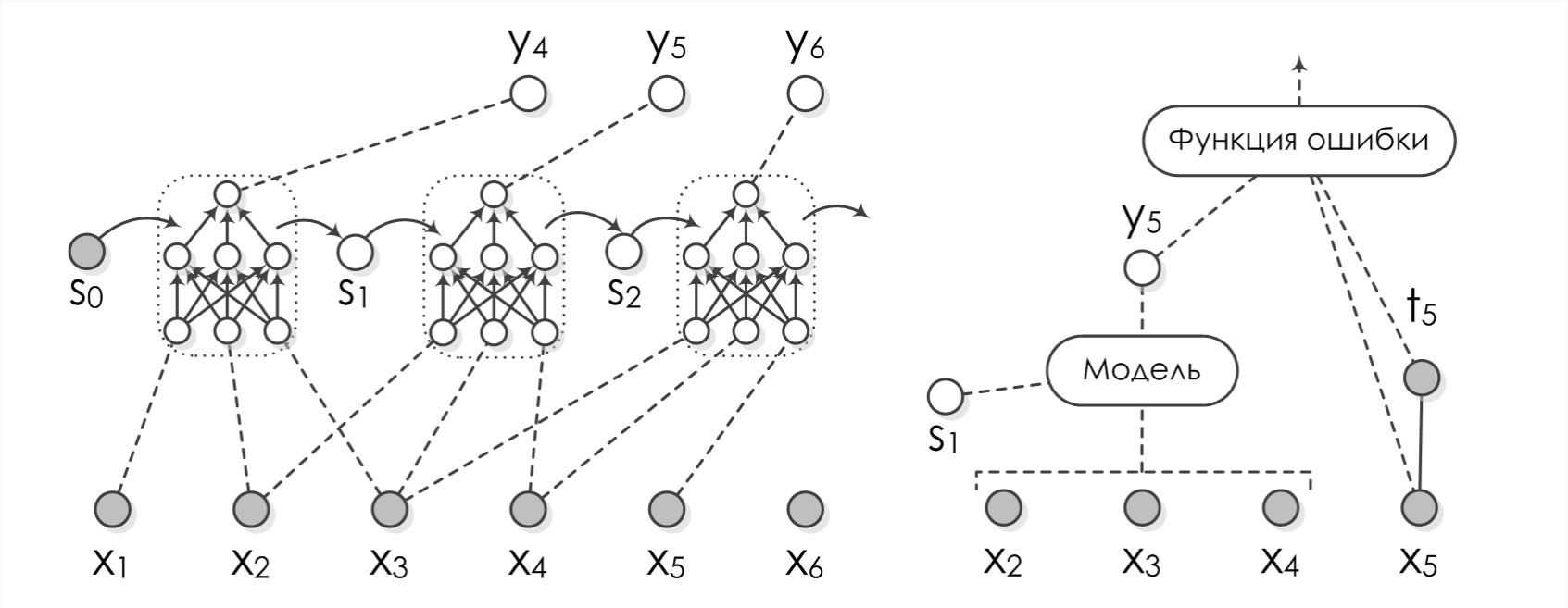
Последовательности

- Как применить к последовательности нейронную сеть?
- Можно использовать скользящее окно:



Последовательности

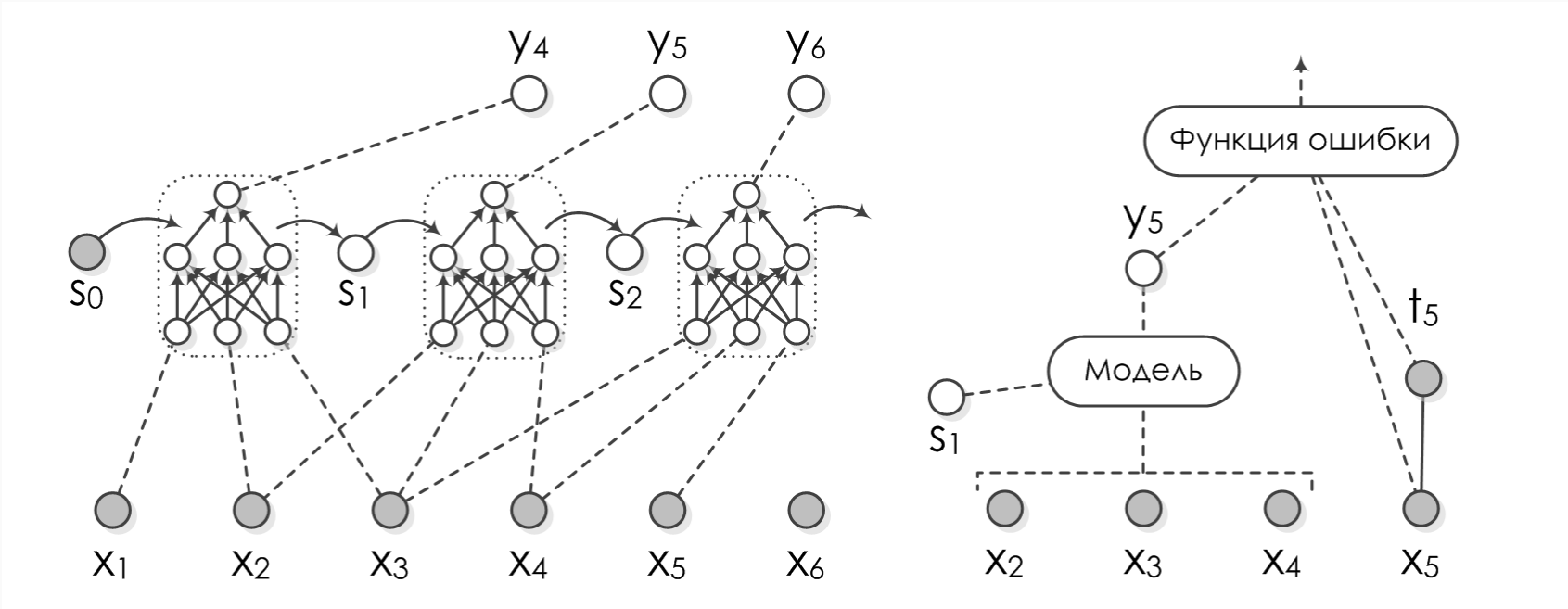
- ...но ещё лучше будет сохранять какое-нибудь скрытое состояние и обновлять его каждый раз.
- Это в точности идея *рекуррентных нейронных сетей* (recurrent neural networks, RNN).



Последовательности

- Но как теперь делать backpropagation? Получается, что в графе вычислений теперь циклы:

$$s_i = h(x_i, x_{i+1}, x_{i+2}, s_{i-1}).$$



- Это же ужасно, и всё сломалось?..

Последовательности

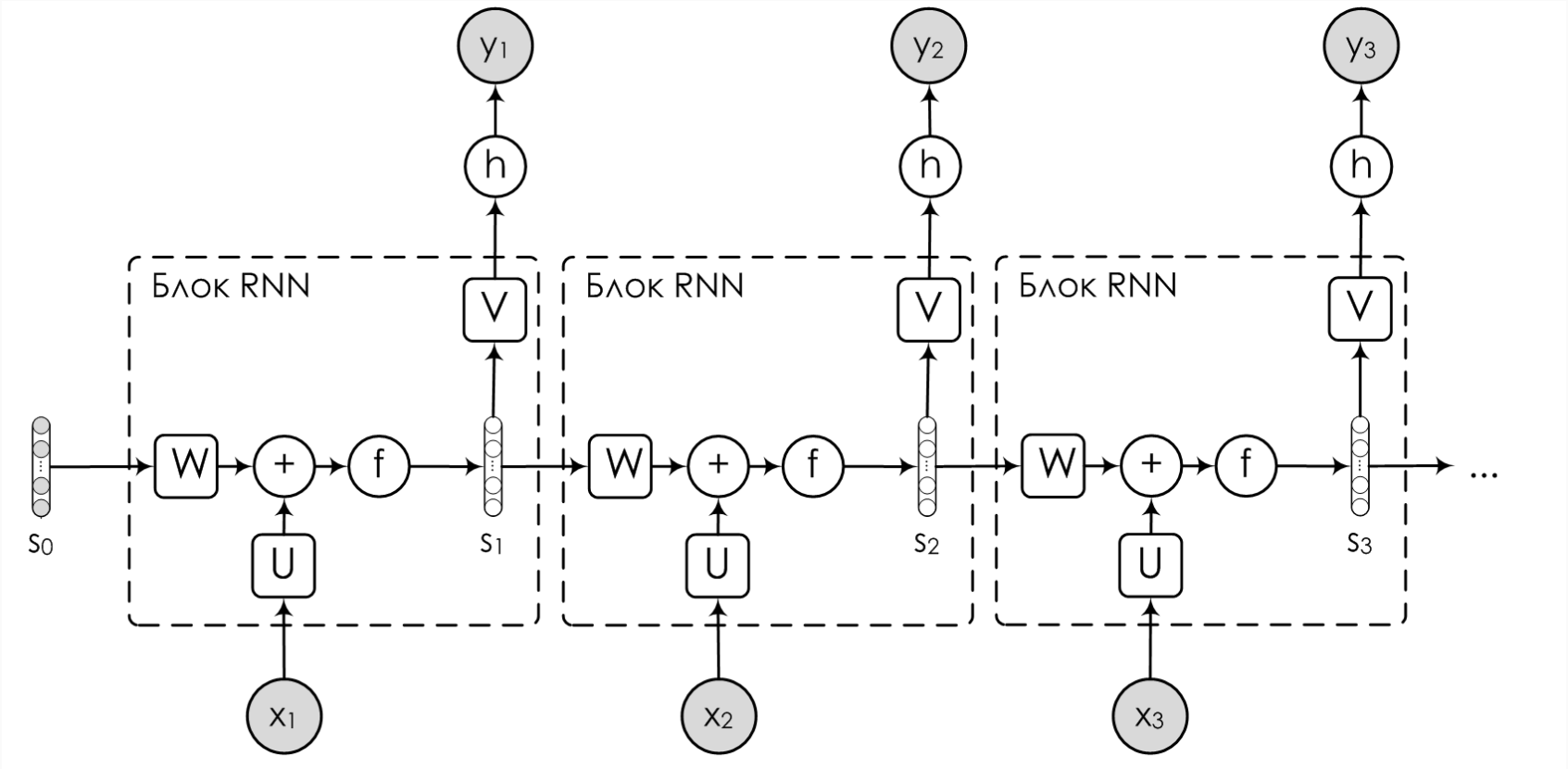
- ...да нет, конечно. Можно “развернуть” циклы обратно:

$$y_6 = f(x_3, x_4, x_5, s_2) = f(x_3, x_4, x_5, h(x_2, x_3, x_4, s_1)) = \\ = f(x_3, x_4, x_5, h(x_2, x_3, x_4, h(x_1, x_2, x_3, s_0))).$$

- Так что формально проблемы нет.
- Но масса проблем в реальности: получается, что рекуррентная сеть – это такая *очень* глубокая сеть с кучей общих весов...

Простая RNN

- “Простая” RNN:



Простая RNN

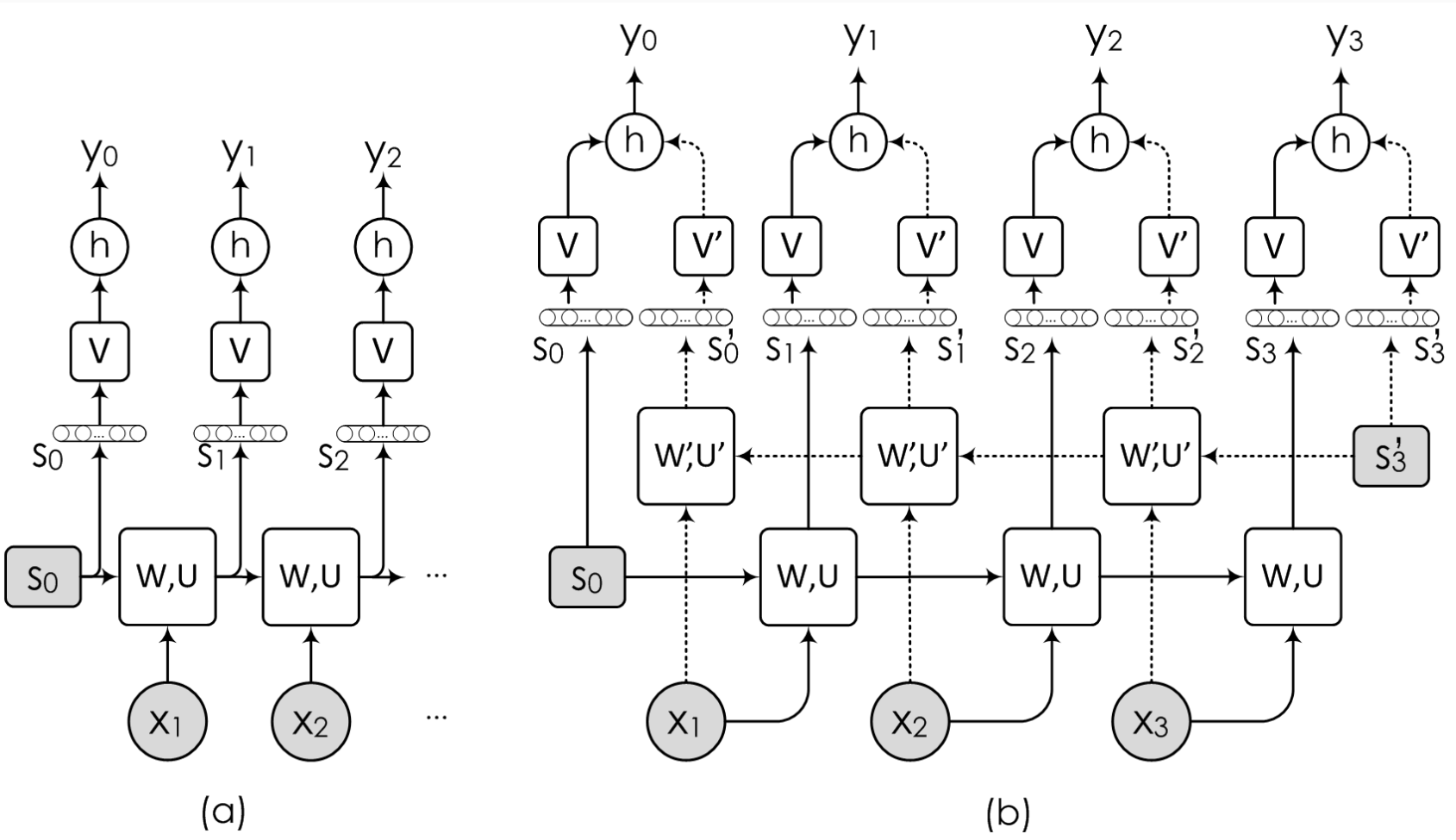
- Формально:

$$\begin{aligned}
\mathbf{a}_t &= \mathbf{b} + W\mathbf{s}_{t-1} + U\mathbf{x}_t, \\
\mathbf{s}_t &= f(\mathbf{a}_t), \\
\mathbf{o}_t &= \mathbf{c} + V\mathbf{s}_t, \\
\mathbf{y}_t &= h(\mathbf{o}_t),
\end{aligned}$$

где f – рекуррентная нелинейность, h – функция выхода.

Двунаправленная RNN

- Иногда нужен контекст с обеих сторон:



Двунаправленная RNN

- Формально:

$$\begin{aligned}
\mathbf{s}_t &= \sigma\left(\mathbf{b} + W\mathbf{s}_{t-1} + U\mathbf{x}_t\right), \\
\mathbf{s}'_t &= \sigma\left(\mathbf{b}' + W'\mathbf{s}'_{t+1} + U'\mathbf{x}_t\right), \\
\mathbf{o}_t &= \mathbf{c} + V\mathbf{s}_t + V'\mathbf{s}'_t, \\
\mathbf{y}_t &= h\left(\mathbf{o}_t\right).
\end{aligned}$$

- И это, конечно, обобщается на любой другой тип конструкций.

Взрывающиеся градиенты

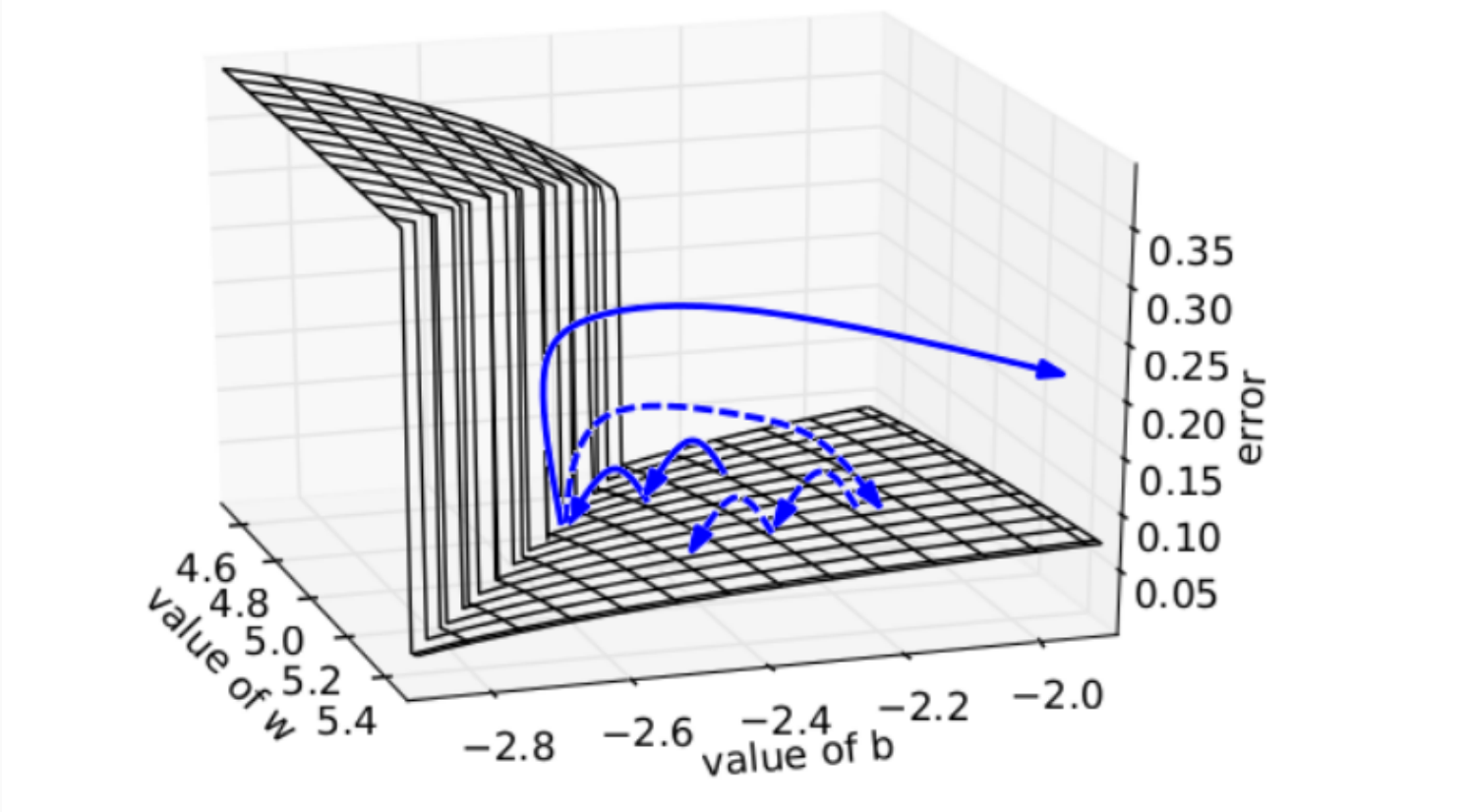
- Две проблемы:
  - взрывающиеся градиенты (exploding gradients);
  - затухающие градиенты (vanishing gradients).
- Надо каждый раз умножать на одну и ту же W, и норма градиента может расти или убывать экспоненциально.
- Взрывающиеся градиенты: надо каждый раз умножать на W, и норма градиента может расти экспоненциально.
- Что делать?

Взрывающиеся градиенты

- Да просто обрезать градиенты, ограничить сверху, чтобы не росли.
- Два варианта – ограничить общую норму или каждое значение:
  - `sgd = optimizers.SGD(lr=0.01, clipnorm=1.)`
  - `sgd = optimizers.SGD(lr=0.01, clipvalue=.05)`

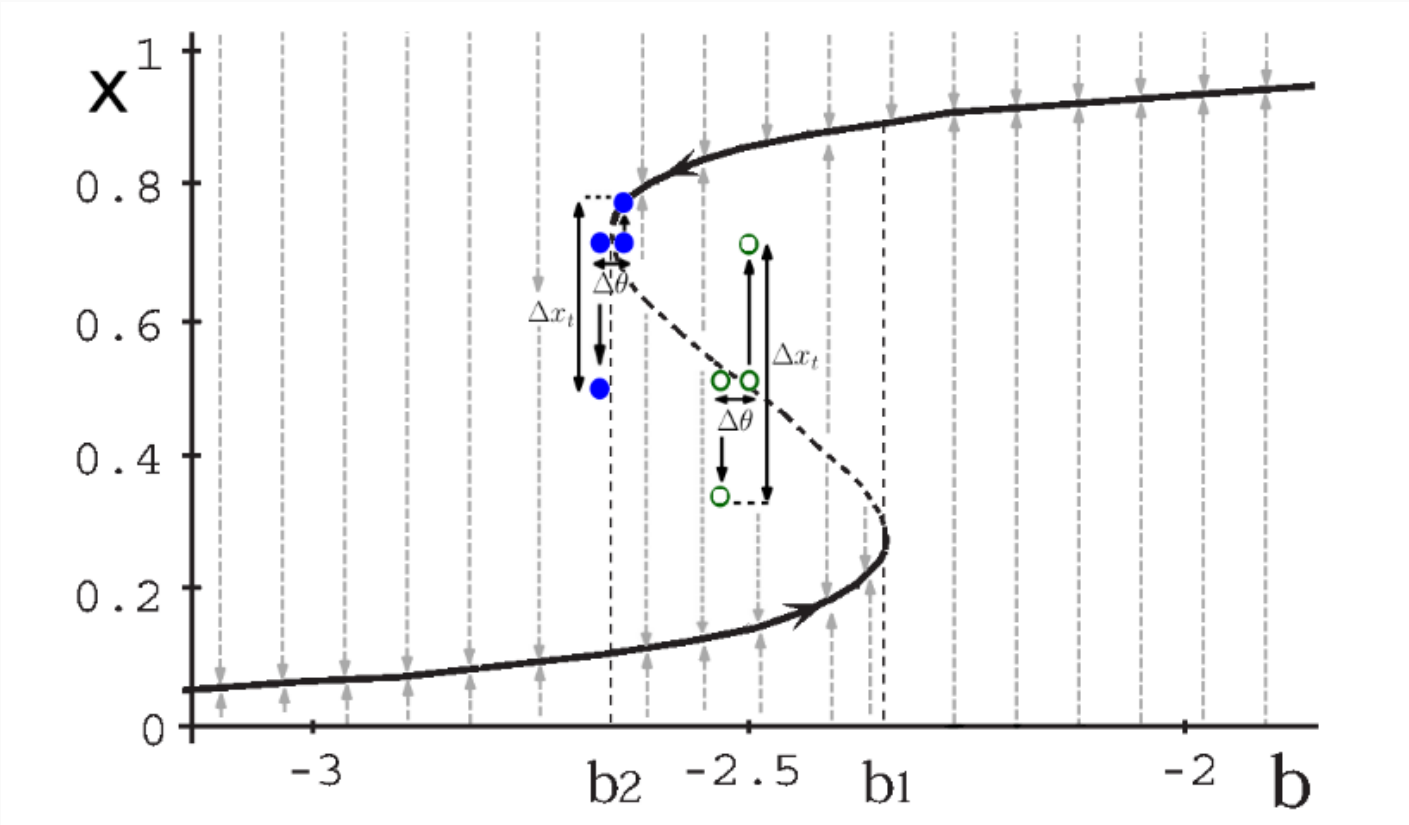
Взрывающиеся градиенты

- (Pascanu et al., 2013) – вот что будет происходить:



Взрывающиеся градиенты

- Там же объясняется, откуда возьмутся такие перепады: есть точки бифуркации у RNN.



Карусель константной ошибки: LSTM и GRU

LSTM

- Затухающие градиенты: надо каждый раз умножать на W.
- Поэтому не получается долгосрочную память реализовать.

- А хочется. Что делать?..

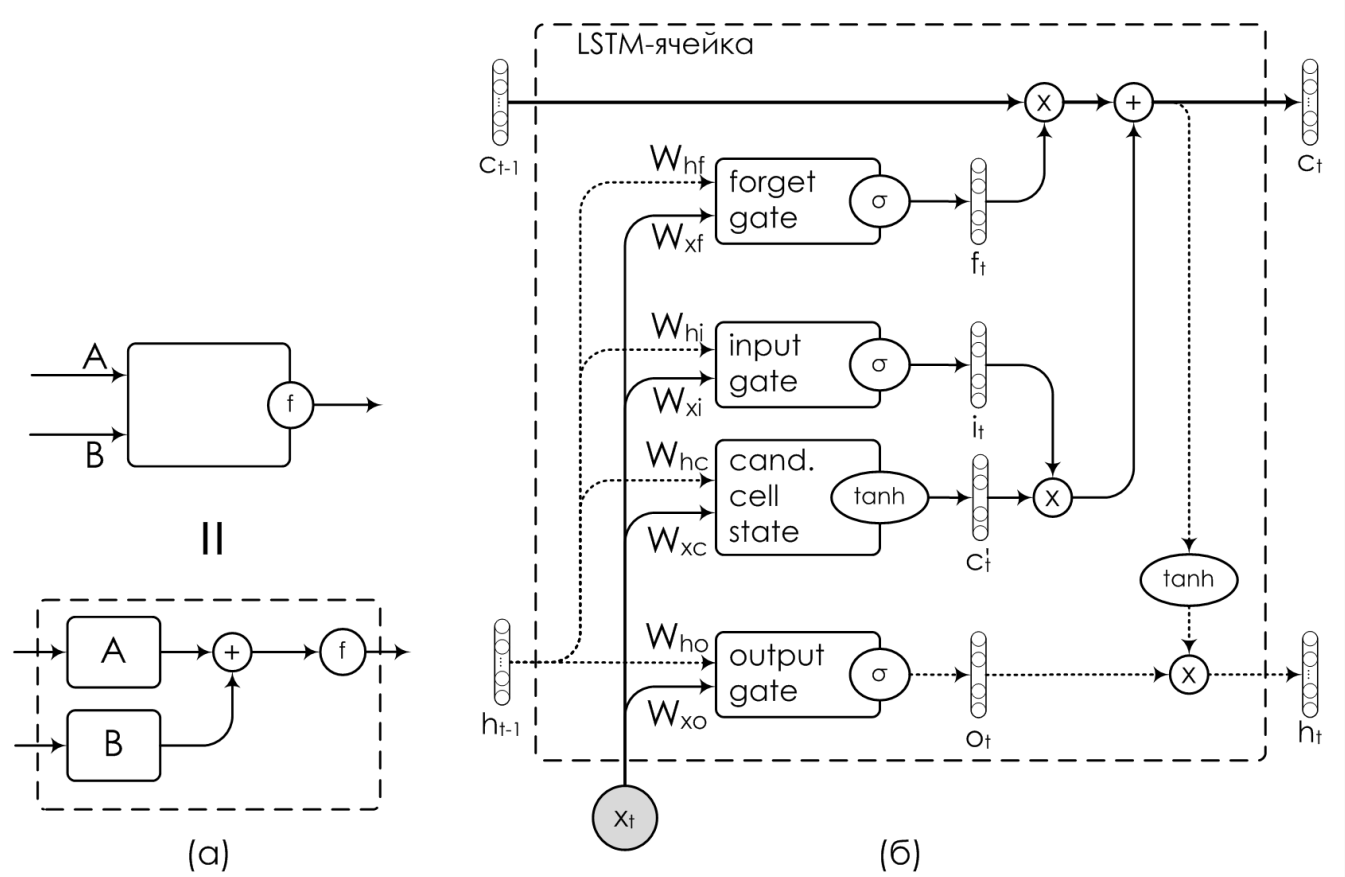
LSTM

- Базовую идею мы уже видели в ResNet: надо сделать так, чтобы градиент проходил.
- В RNN это называется «карусель константной ошибки» (constant error carousel).

- Идея из середины 1990-х (Шмидхубер): давайте составлять RNN из более сложных частей, в которых будет прямой путь для градиентов, и память будет контролироваться явно.

LSTM

- LSTM (long short-term memory). “Ванильный” LSTM: $\mathbf{c}_t$ – состояние ячейки памяти, $\mathbf{h}_t$ – скрытое состояние.
- Input gate и forget gate определяют, надо ли менять $\mathbf{c}_t$ на нового кандидата в состояния ячейки.



LSTM

- Формально:

$$
\begin{aligned}
\mathbf{c}'_t &= \tanh\left(W_{xc}\mathbf{x}_t + W_{hc}\mathbf{h}_{t-1} + \mathbf{b}_{c'}\right) && \textit{candidate cell state} \\
\mathbf{i}_t &= \sigma\left(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i\right) && \textit{input gate} \\
\mathbf{f}_t &= \sigma\left(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f\right) && \textit{forget gate} \\
\mathbf{o}_t &= \sigma\left(W_{xo}\mathbf{x}_t + W_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o\right) && \textit{output gate} \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{c}'_t, && \textit{cell state} \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t) && \textit{block output}
\end{aligned}
$$

- Так что LSTM может контролировать состояние ячейки при помощи скрытого состояния и весов.
- Например, если forget gate закрыт ($\mathbf{f}_t = 1$), то получится карусель константной ошибки: $\mathbf{c}_t = \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{c}'_t$, и $\frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}} = 1$.
- Важно инициализировать $\mathbf{b}_f$ большим, чтобы forget gate был закрыт поначалу.

LSTM

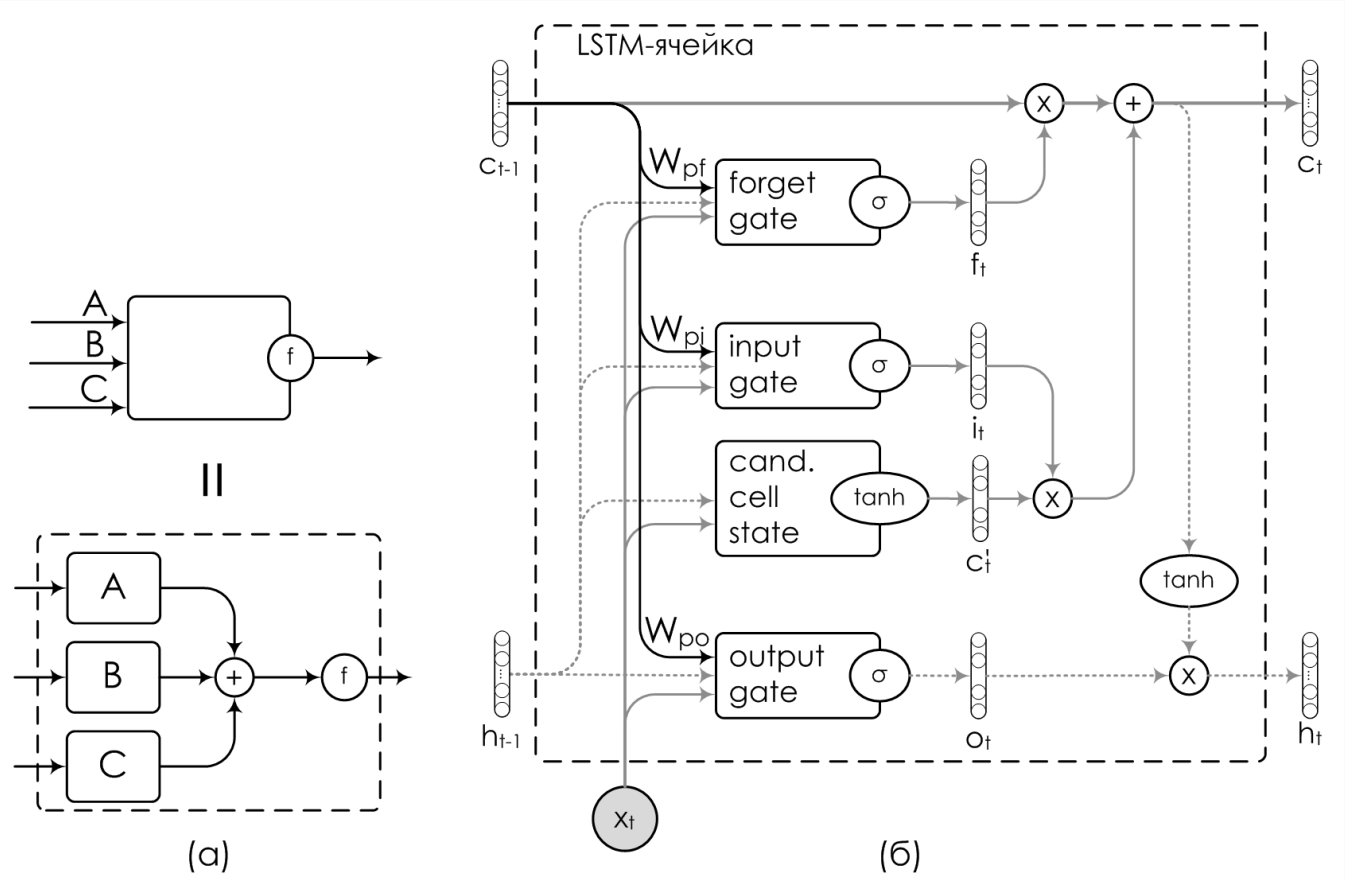
- LSTM был создан в середине 1990-х (Hochreiter and Schmidhuber, 1995; 1997).
- В полностью современной форме в (Gers, Schmidhuber, 2000).
- Проблема: хотим управлять $\mathbf{c}$, но гейты его не получают! Они видят только $\mathbf{h}_{t-1}$, а это

$$\mathbf{h}_{t-1} = \mathbf{o}_{t-1} \odot \tanh(\mathbf{c}_{t-1}).$$

- Так что если output gate закрыт, то поведение LSTM вообще от состояния ячейки не зависит.
- Нехорошо. Что делать?..

LSTM

- …конечно, добавить ещё несколько матриц! (peepholes)



LSTM

- Формально:

$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + W_{pi}\mathbf{c}_{t-1} + \mathbf{b}_i\right) \\
\mathbf{f}_t &= \sigma\left(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + W_{pf}\mathbf{c}_{t-1} + \mathbf{b}_f\right) \\
\mathbf{o}_t &= \sigma\left(W_{xo}\mathbf{x}_t + W_{ho}\mathbf{h}_{t-1} + W_{po}\mathbf{c}_{t-1} + \mathbf{b}_o\right)
\end{aligned}$$

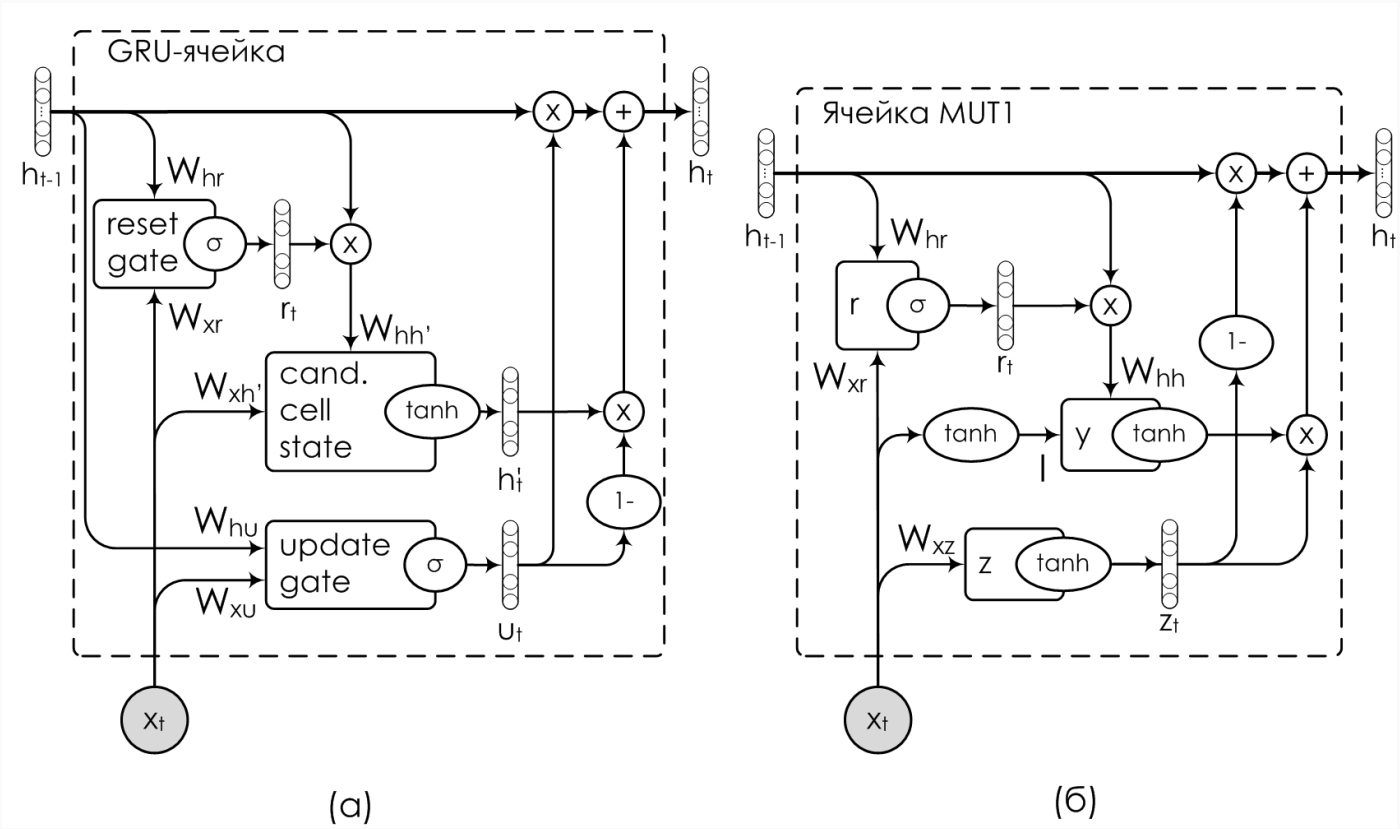
- Видно, что тут есть огромное поле для вариантов LSTM: можно удалить любой гейт, любую замочную скважину, поменять функции активации...
- Как выбрать?

LSTM

- «LSTM: a Search Space Odyssey» (Greff et al., 2015).
- Большое экспериментальное сравнение.
- В честности, некоторые куда более простые архитектуры (без одного из гейтов!) не сильно проигрывали «ванильному» LSTM.
- И это приводит нас к...

GRU

- ...Gated Recurrent Units (GRU; Cho et al., 2014).
- В GRU тоже есть прямой путь для градиентов, но проще.



GRU

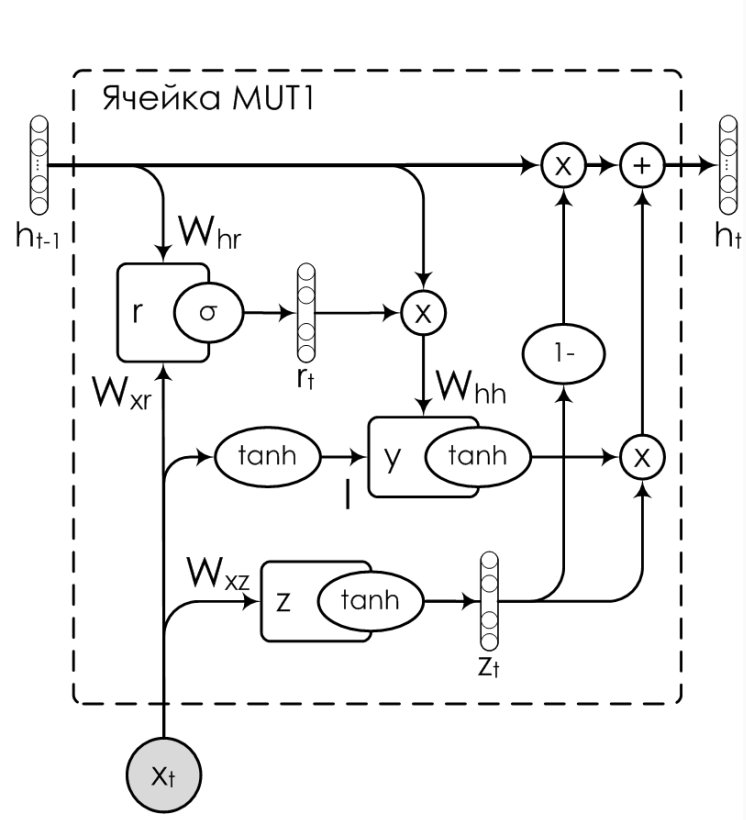
- Формально:

$$
\begin{aligned}
u_t &= \sigma(W_{xu}\mathbf{x}_t + W_{hu}\mathbf{h}_{t-1} + \mathbf{b}_u) \\
\mathbf{r}_t &= \sigma(W_{xr}\mathbf{x}_t + W_{hr}\mathbf{h}_{t-1} + \mathbf{b}_r) \\
\mathbf{h}'_t &= \tanh(W_{xh'}\mathbf{x}_t + W_{hh'}(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \\
\mathbf{h}_t &= (1 - u_t) \odot \mathbf{h}'_t + u_t \odot \mathbf{h}_{t-1}
\end{aligned}
$$

- Теперь есть update gate и reset gate, нет разницы между $\mathbf{c}_t$ и $\mathbf{h}_t$.
- Меньше матриц (6, а не 8 или 11 с замочными скважинами), меньше весов, но только чуть хуже LSTM работает.
- Так что можно больше GRU поместить, и сеть станет лучше.

GRU

- Другие варианты тоже есть.
- (Józefowicz, Zaremba, Sutskever, 2015): огромное сравнение, выращивали архитектуры эволюционными методами.
- Три новых интересных архитектуры; например:



Спасибо!

Спасибо за внимание!