Data Mining in Action

Лекция 10. Рекуррентные нейросети

Партнеры курса

МИСиС
Национальный исследовательский технологический университет

misis.ru

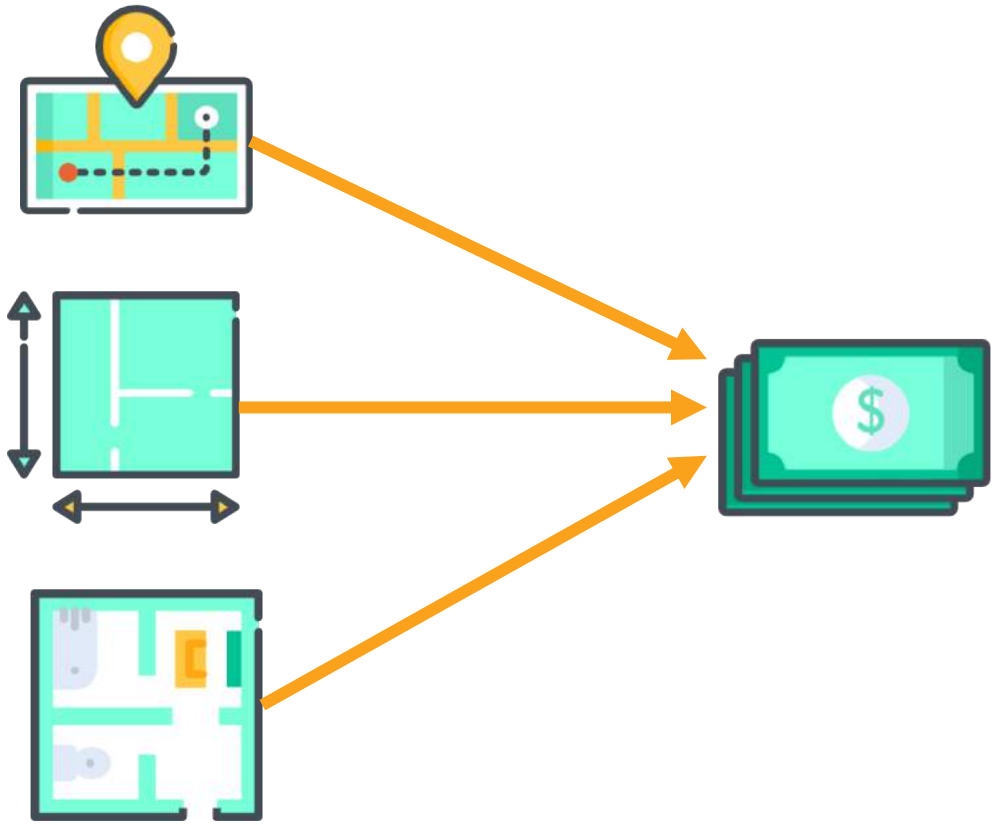
jet.su

План

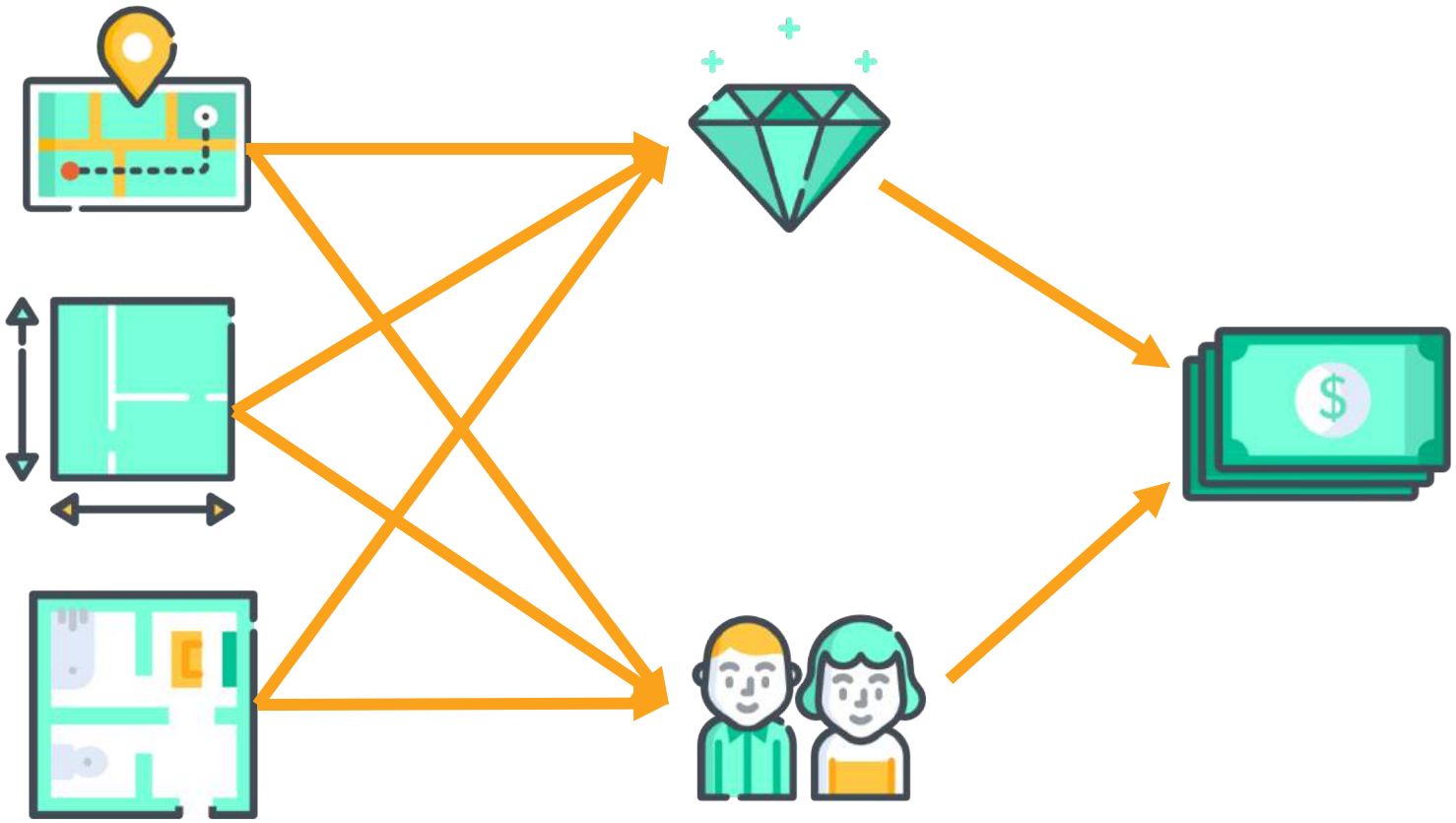
1. Напоминание
2. Vanilla RNN
3. LSTM и GRU
4. Seq2seq & attention

1. Напоминание

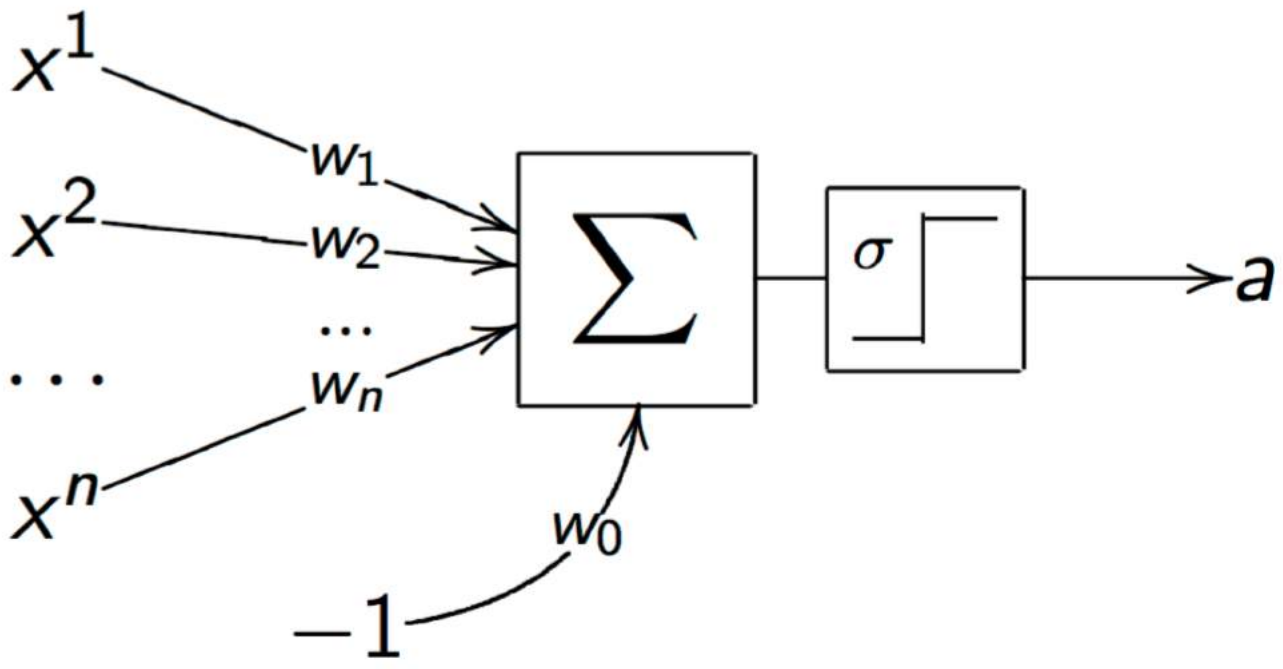
Пример: прогнозирование цен на недвижимость



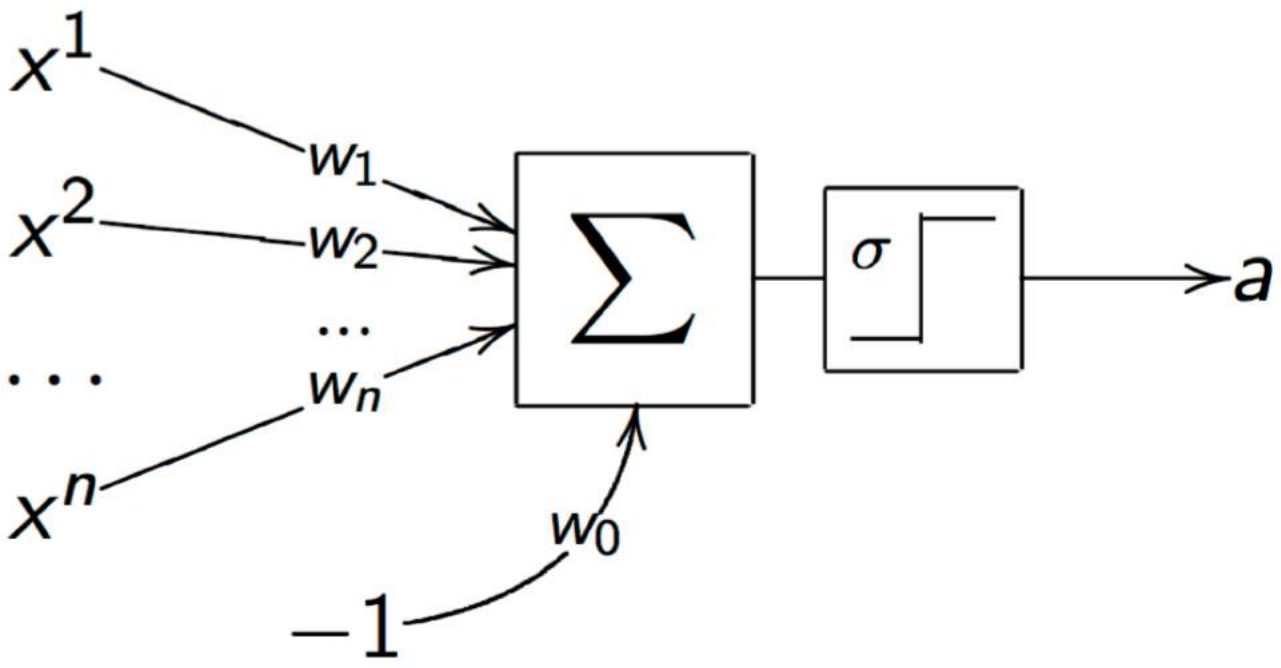
Пример: прогнозирование цен на недвижимость



Нейрон в машинном обучении

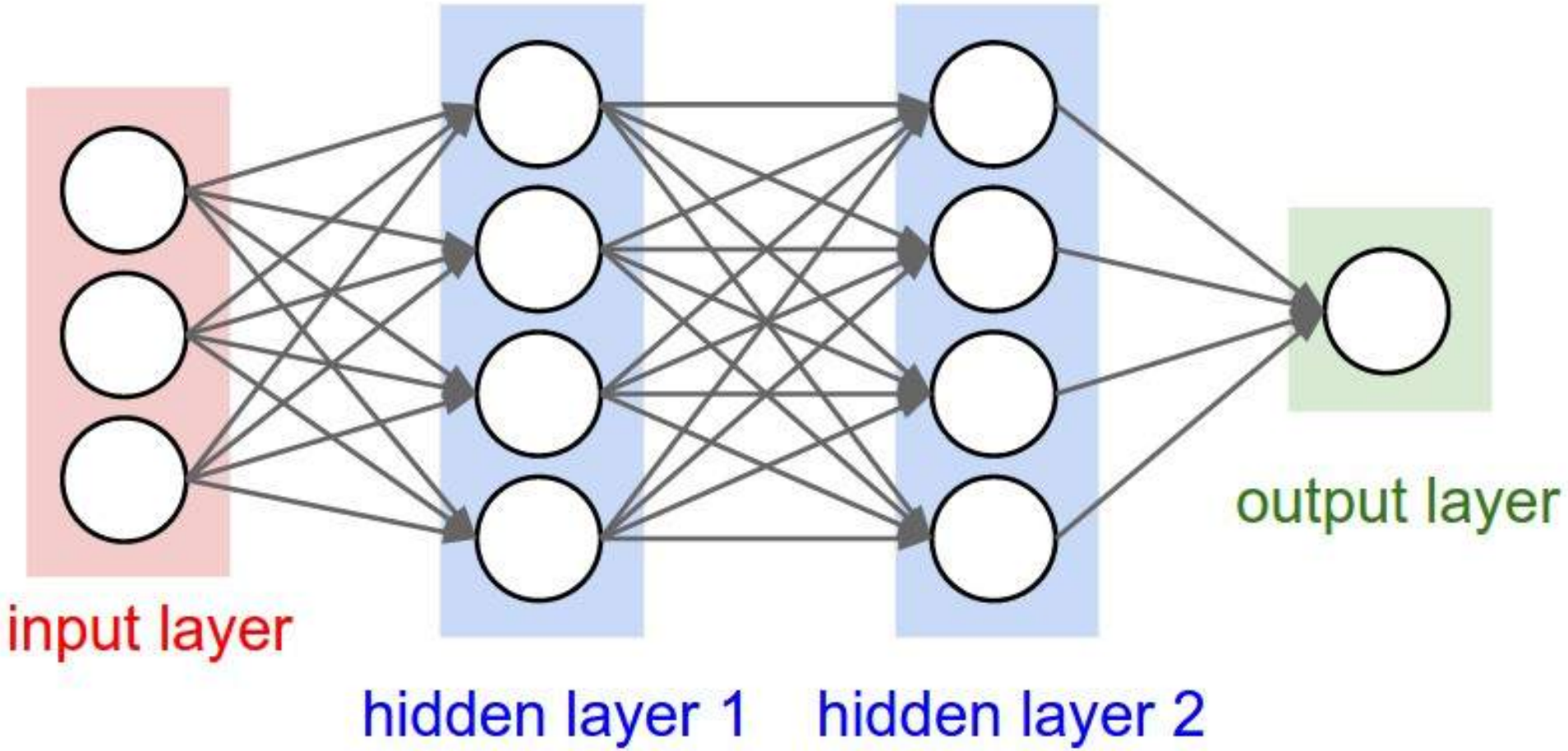


Модель нейрона

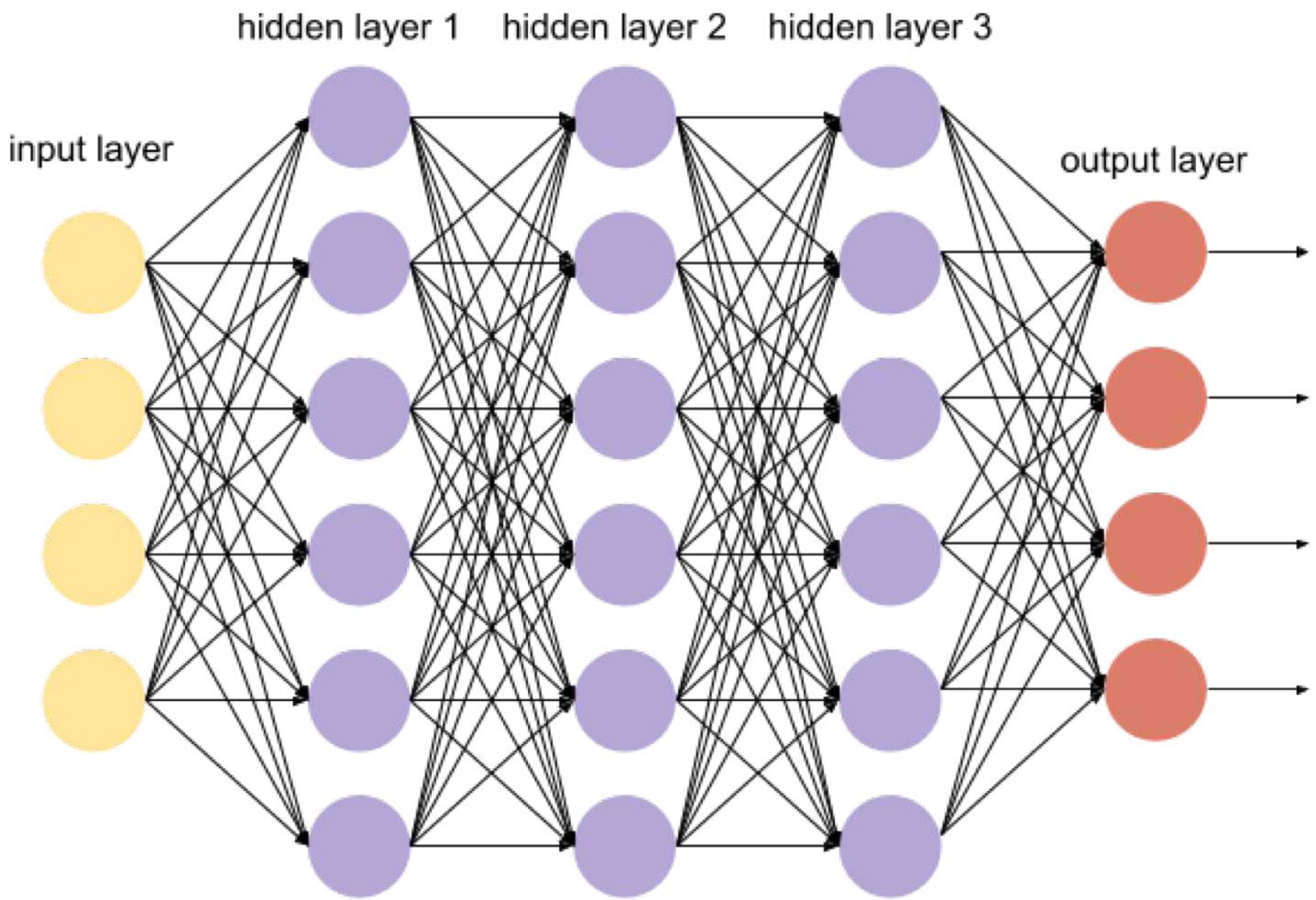


$$a(x) = \sigma(w^T x + b)$$

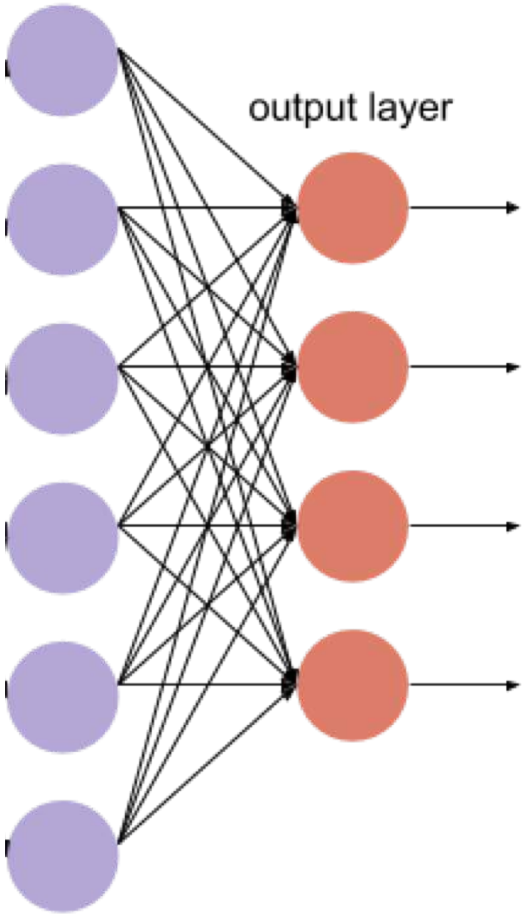
Нейронная сеть (один выход)



Нейронная сеть (много выходов)

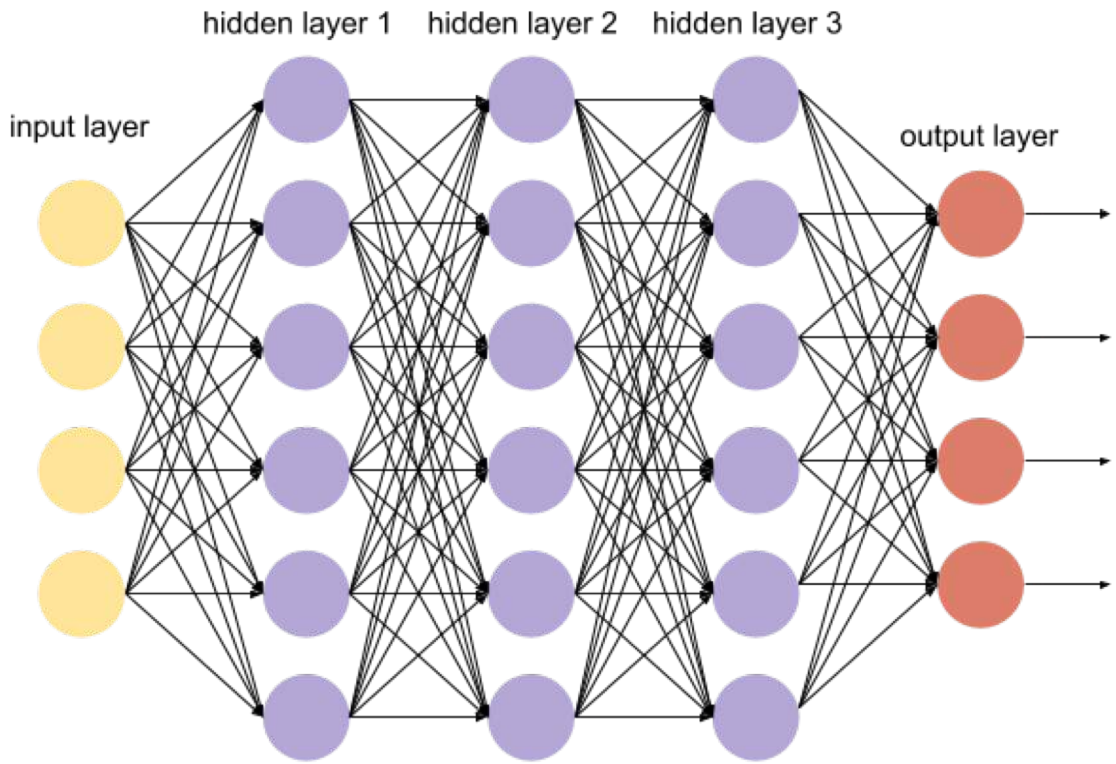


Если в следующем слое много нейронов



$$h_t = f(W h_{t-1} + b)$$

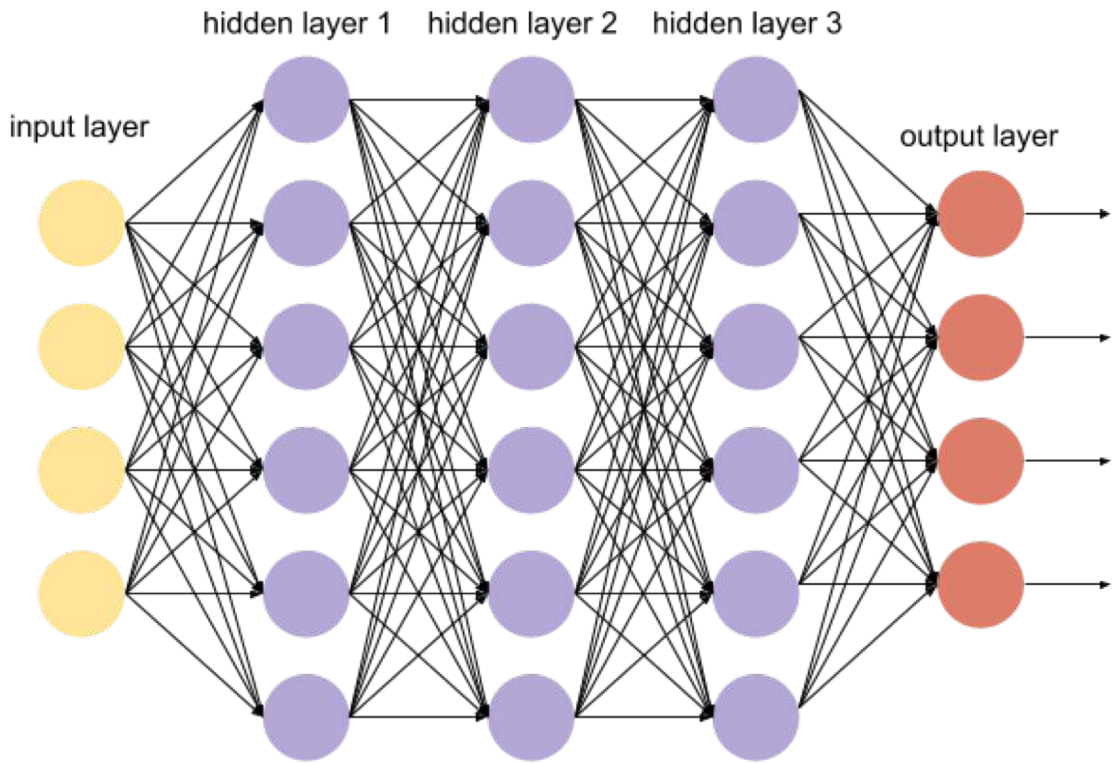
Нейронная сеть



$$h_t = f(W h_{t-1} + b)$$

$$f_0(W_0 x + b_0)$$

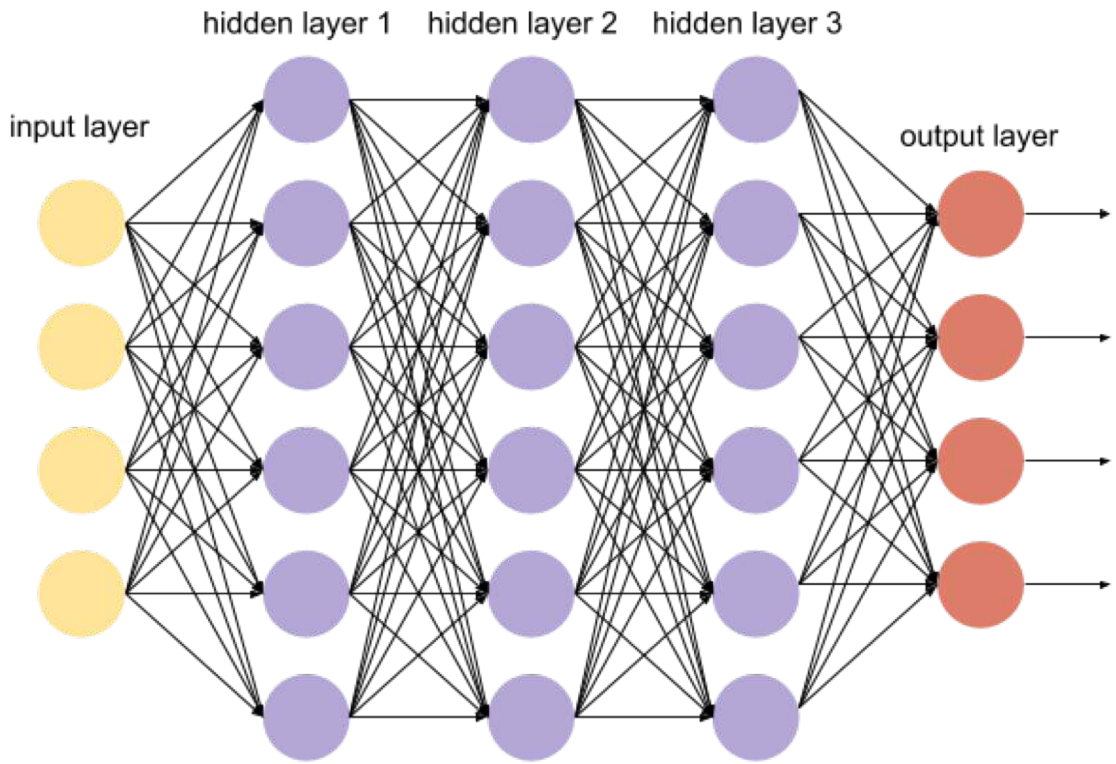
Нейронная сеть



$$h_t = f(W h_{t-1} + b)$$

$$f_1(W_1 f_0(W_0 x + b_0) + b_1)$$

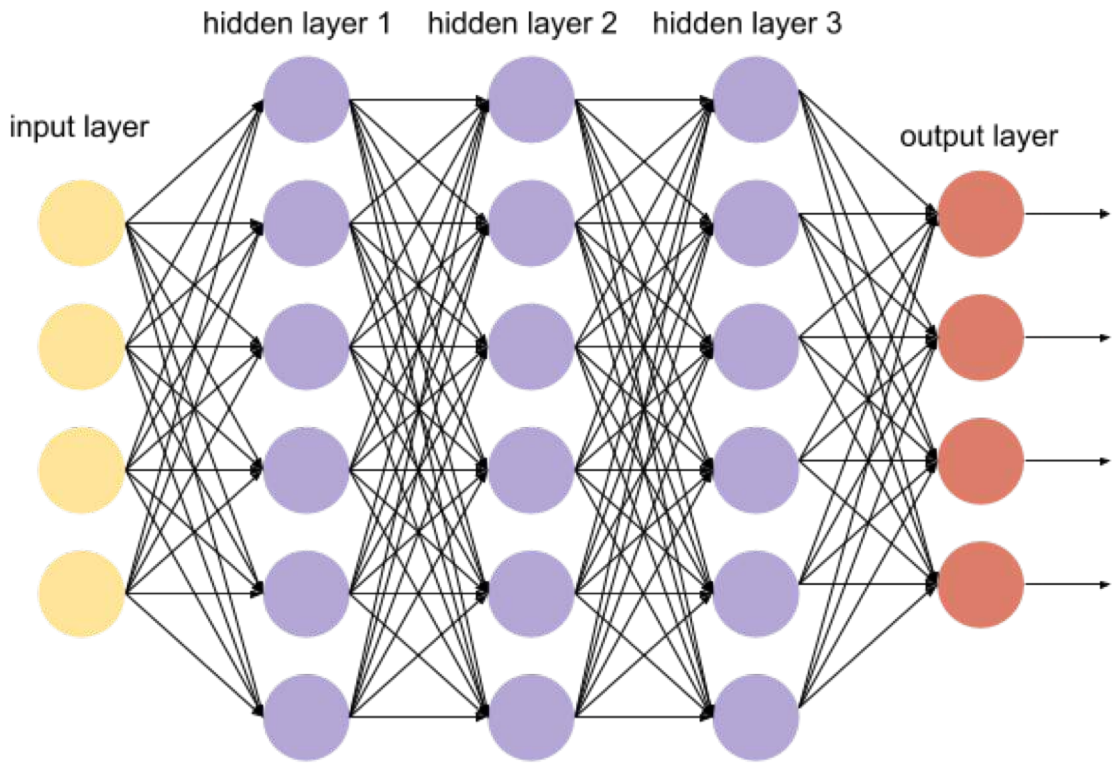
Нейронная сеть



$$h_t = f(W h_{t-1} + b)$$

$$f_2(W_2 f_1(W_1 f_0(W_0 x + b_0) + b_1) + b_2)$$

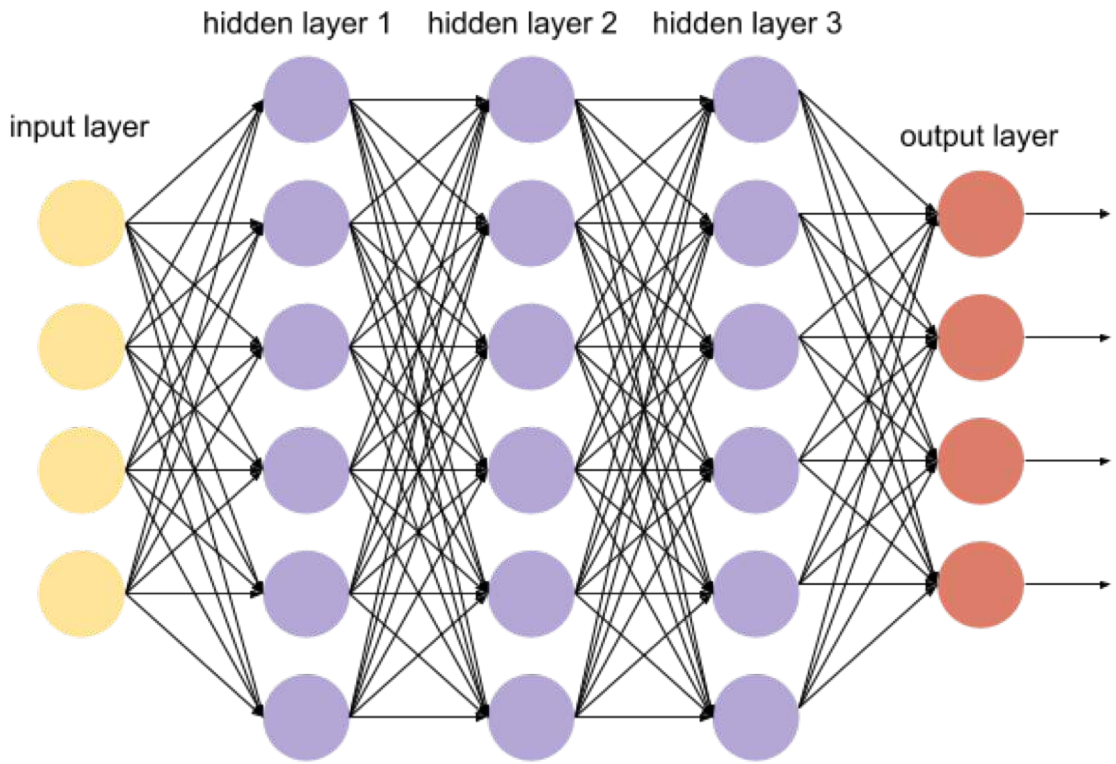
Нейронная сеть



$$h_t = f(W h_{t-1} + b)$$

$$f_3(W_3 f_2(W_2 f_1(W_1 f_0(W_0 x + b_0) + b_1) + b_2) + b_3)$$

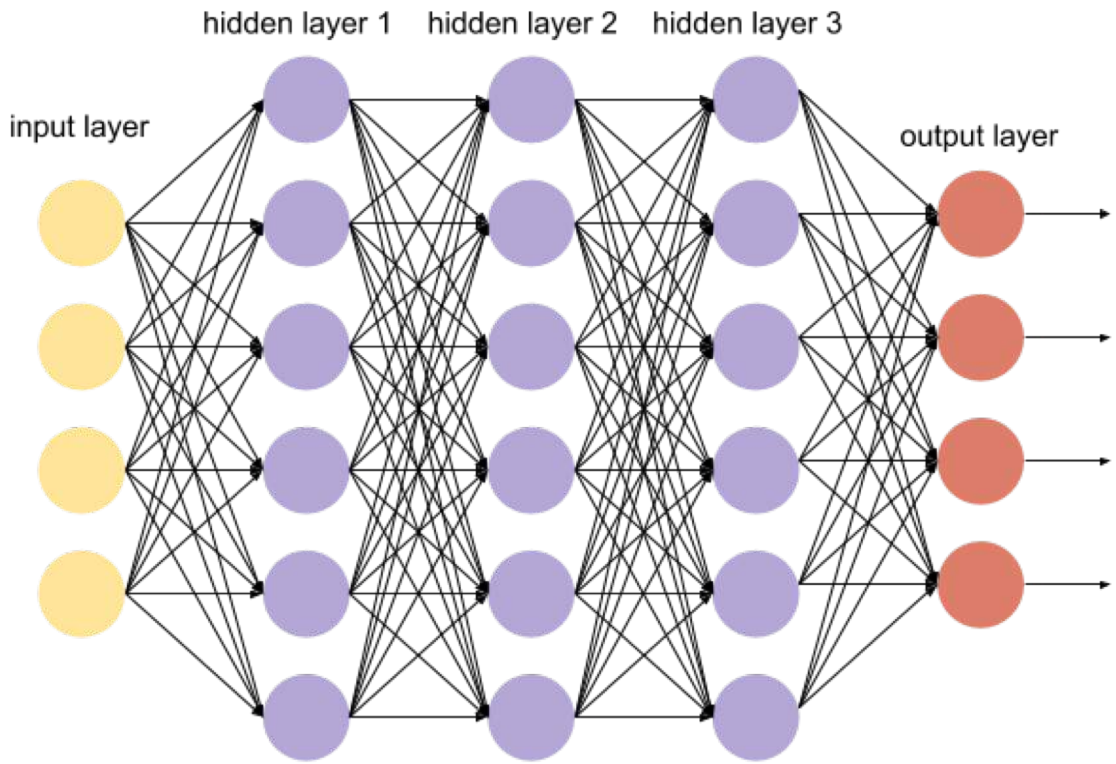
Нейронная сеть



$$h_t = f(W h_{t-1} + b)$$

$$a(x) = f_3(W_3 f_2(W_2 f_1(W_1 f_0(W_0 x + b_0) + b_1) + b_2) + b_3)$$

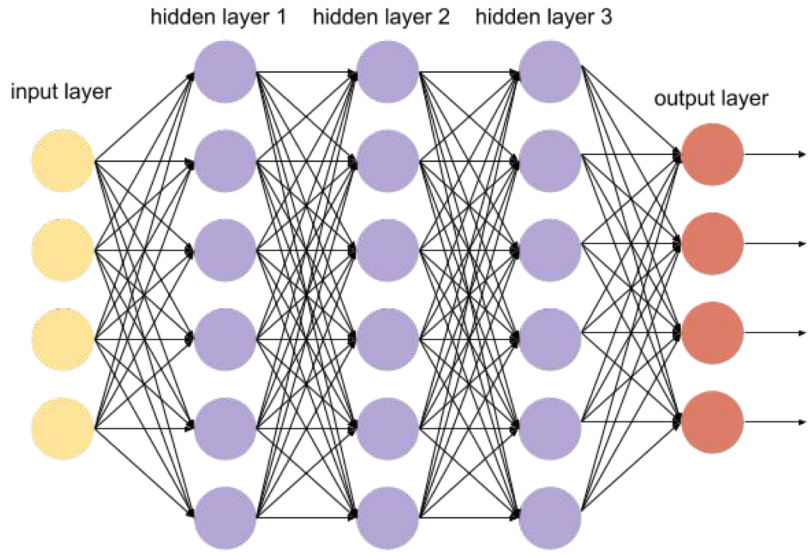
Нейронная сеть



$$h_t = f(W h_{t-1} + b)$$

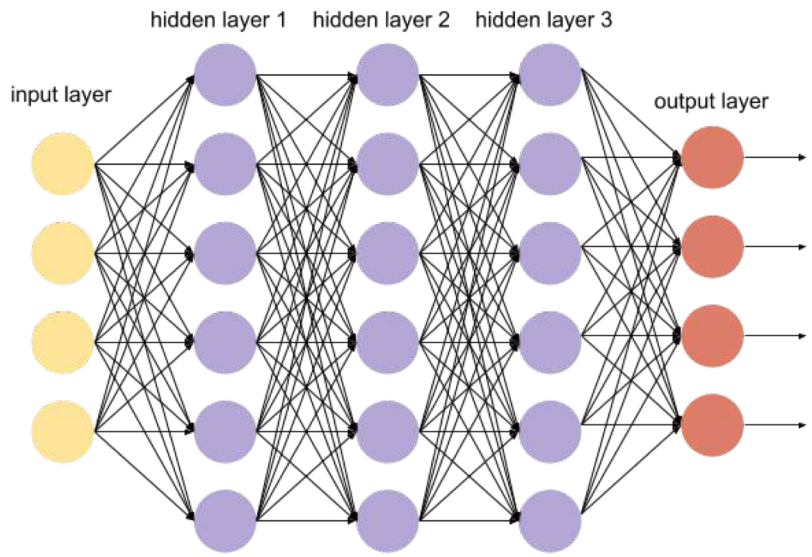
$$a(x) = f_3(W_3 f_2(W_2 f_1(W_1 f_0(W_0 x + b_0) + b_1) + b_2) + b_3)$$

Обучение сети



Задача обучения – настроить веса связей между нейронами на основе обучающей выборки

Обучение сети

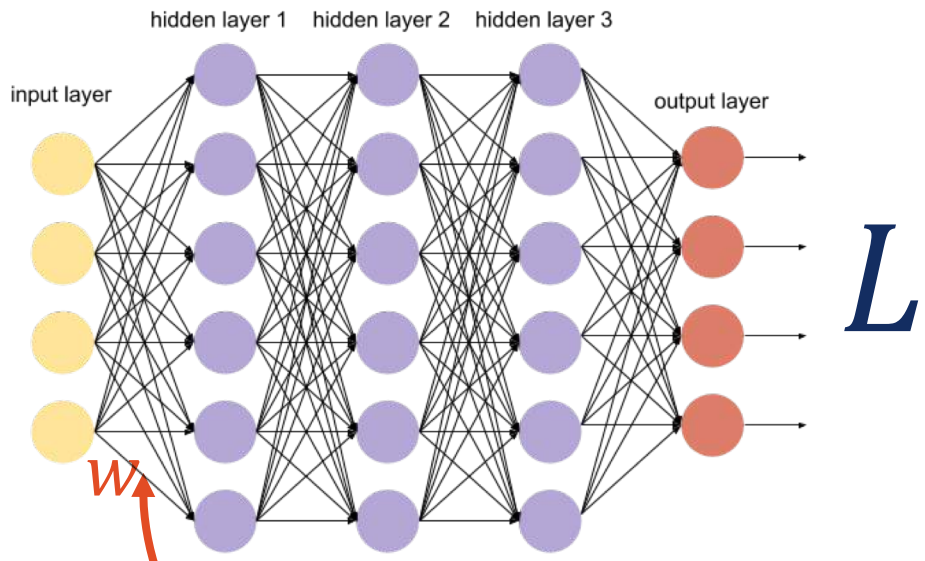


L

Задача обучения – настроить веса связей между нейронами на основе обучающей выборки

1. Выбираем функцию потерь

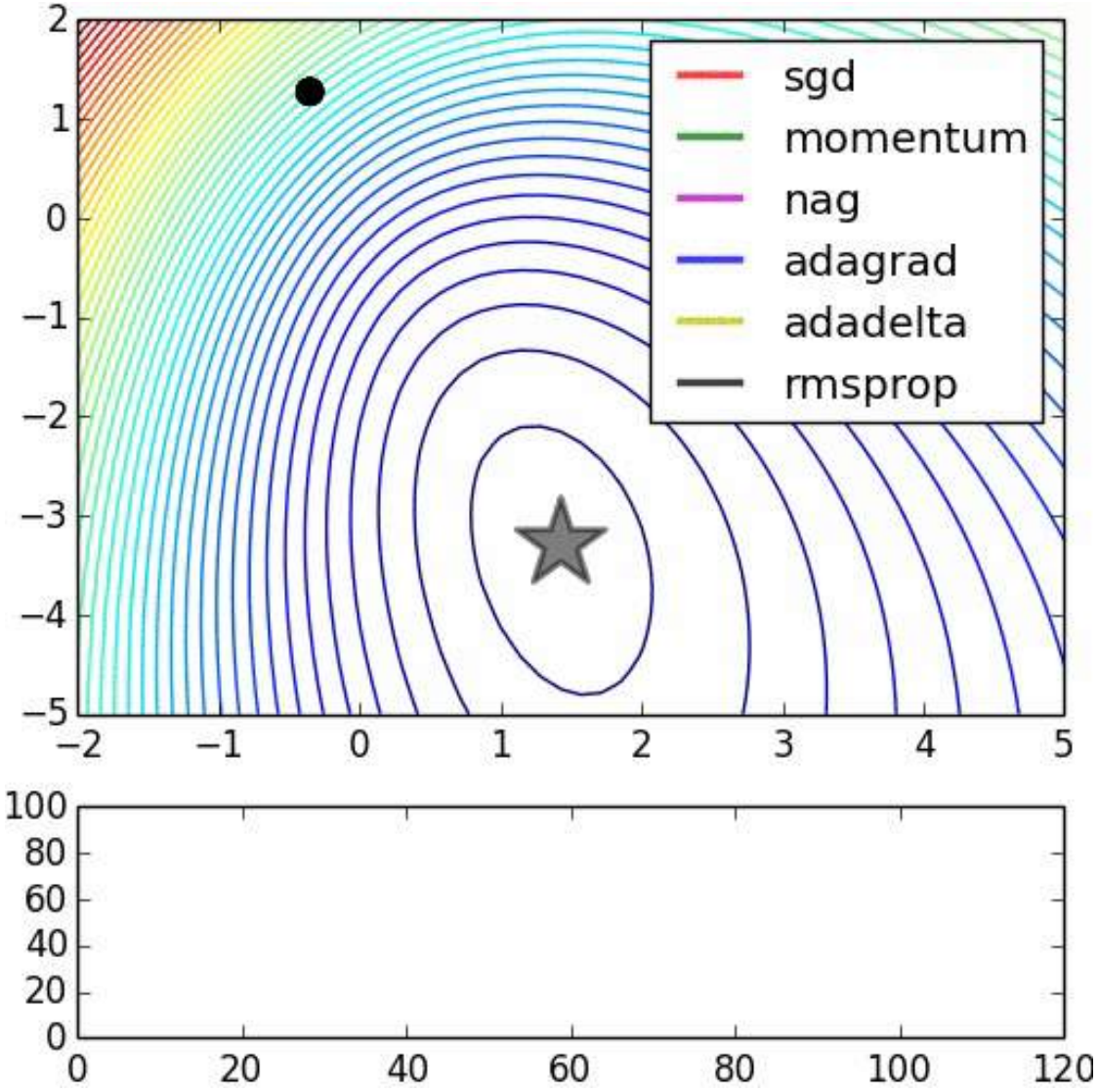
Обучение сети



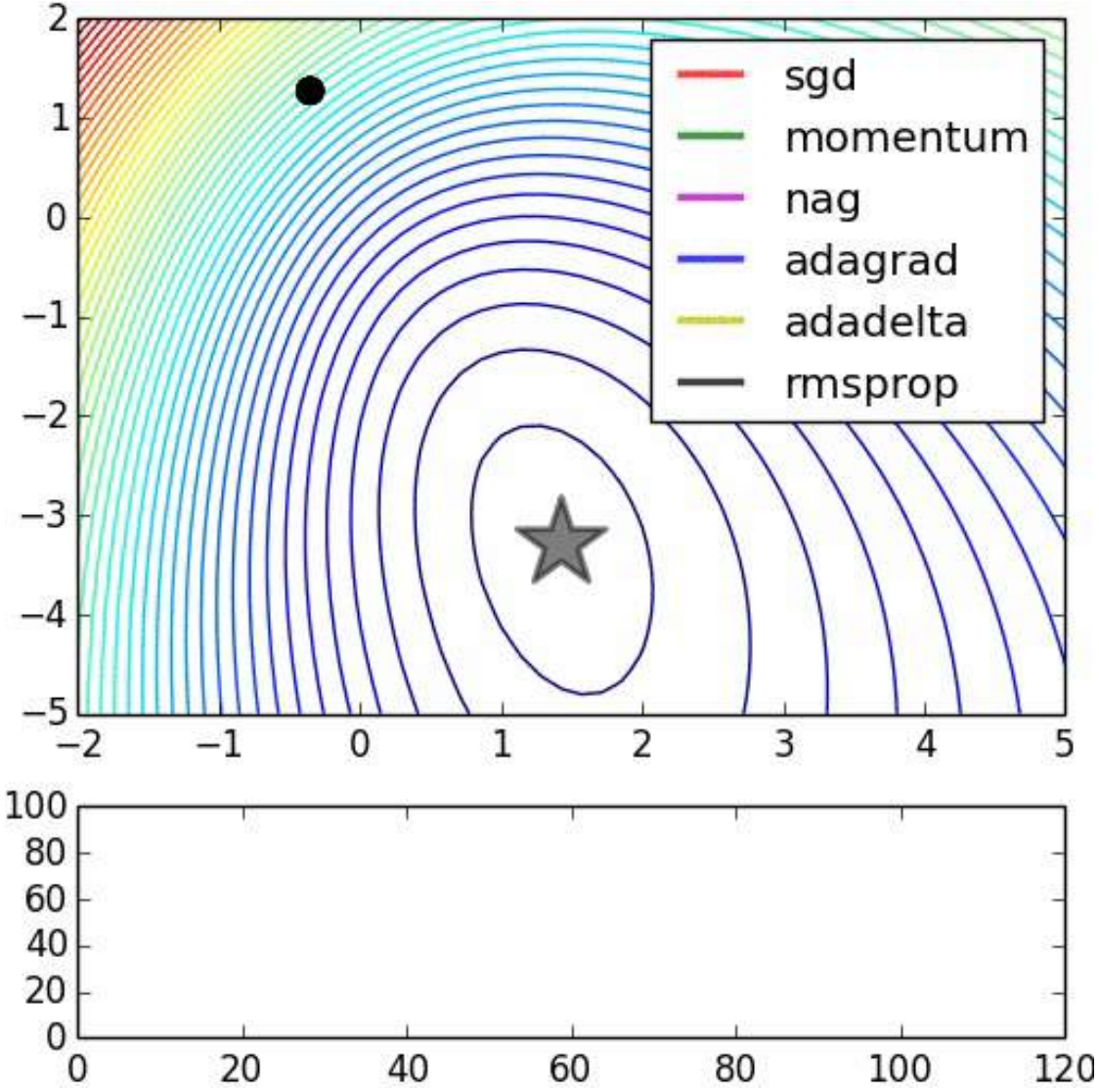
Задача обучения – настроить веса связей между нейронами на основе обучающей выборки

1. Выбираем функцию потерь
2. Обучаем веса с помощью SGD

Модификации SGD



Модификации SGD

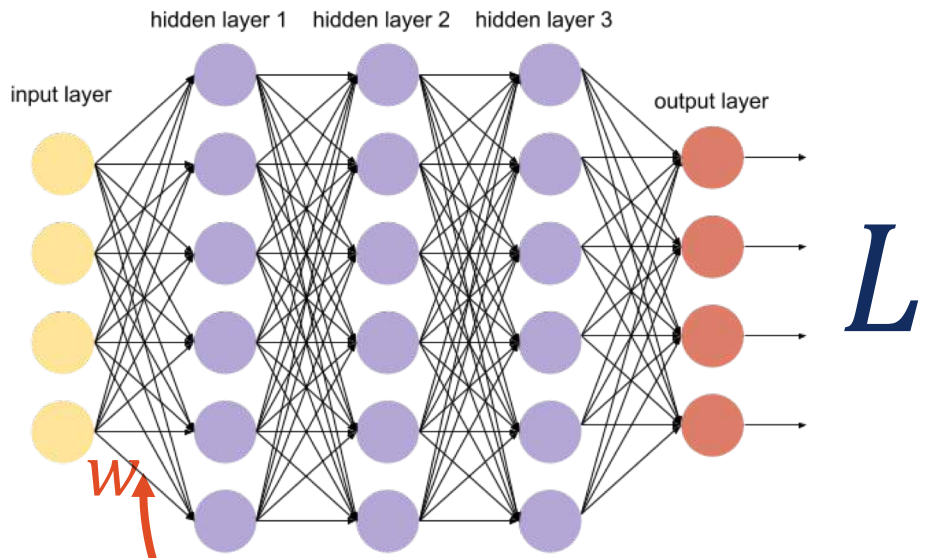


Momentum:
Запоминает направление предыдущего шага и с небольшим весом учитывает его в текущем (движение «по инерции»)

NAG (Nesterov Accelerated Gradient):
Модификация Momentum: антиградиент берем в той точки, куда шагнули бы по инерции

Adagrad, Adadelta, RMSprop, Adam:
Добавлены эвристики для настройки разного шага по разным координатам (весам)

Обучение сети



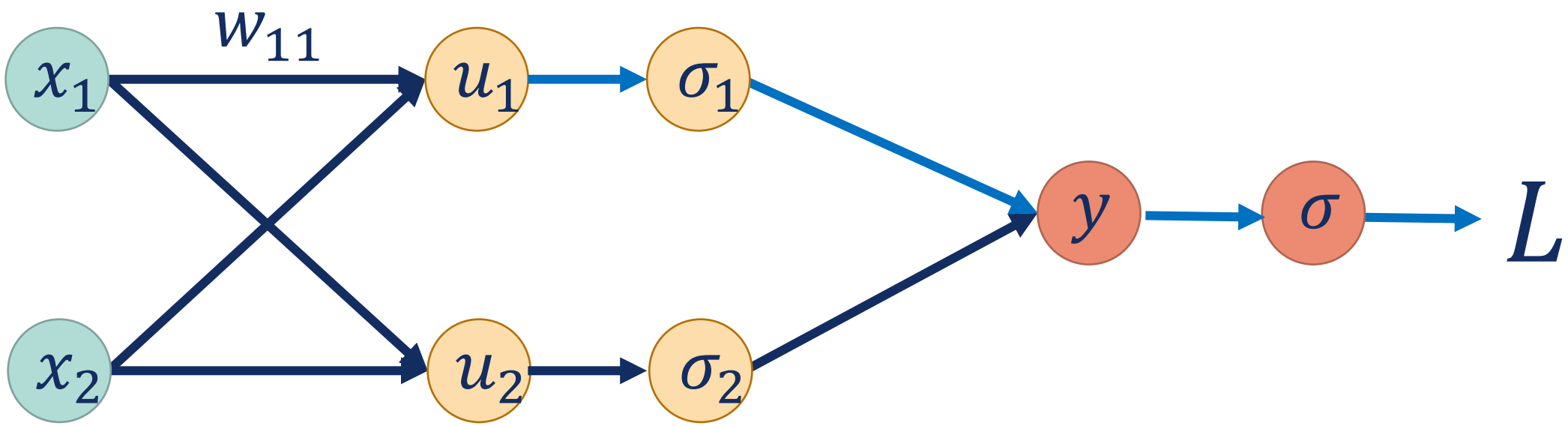
$$L$$

$$w^{(t+1)} = w^{(t)} - \gamma_t \frac{\partial L}{\partial w}$$

Задача обучения – настроить веса связей между нейронами на основе обучающей выборки

1. Выбираем функцию потерь и записываем ошибку сети
2. Обучаем веса с помощью SGD

Граф вычислений для нейросети

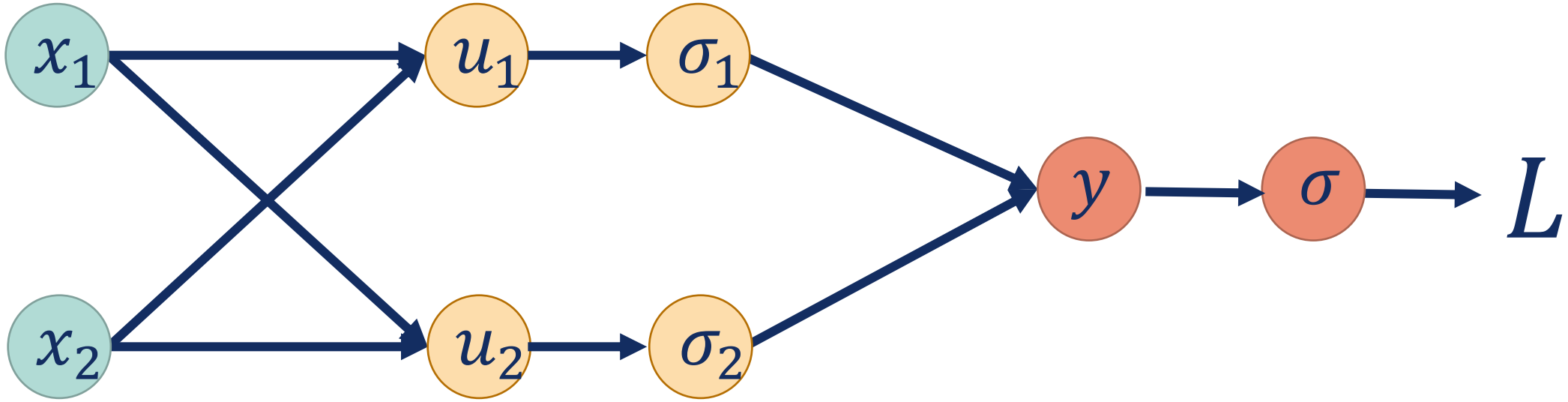


$$w_{11}^{(t+1)} = w_{11}^{(t)} - \gamma_t \frac{\partial L}{\partial w_{11}}$$

$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial u_1}\frac{\partial u_1}{\partial w_{11}} = \frac{\partial L}{\partial u_1} x_1$$

Backprop: как делать

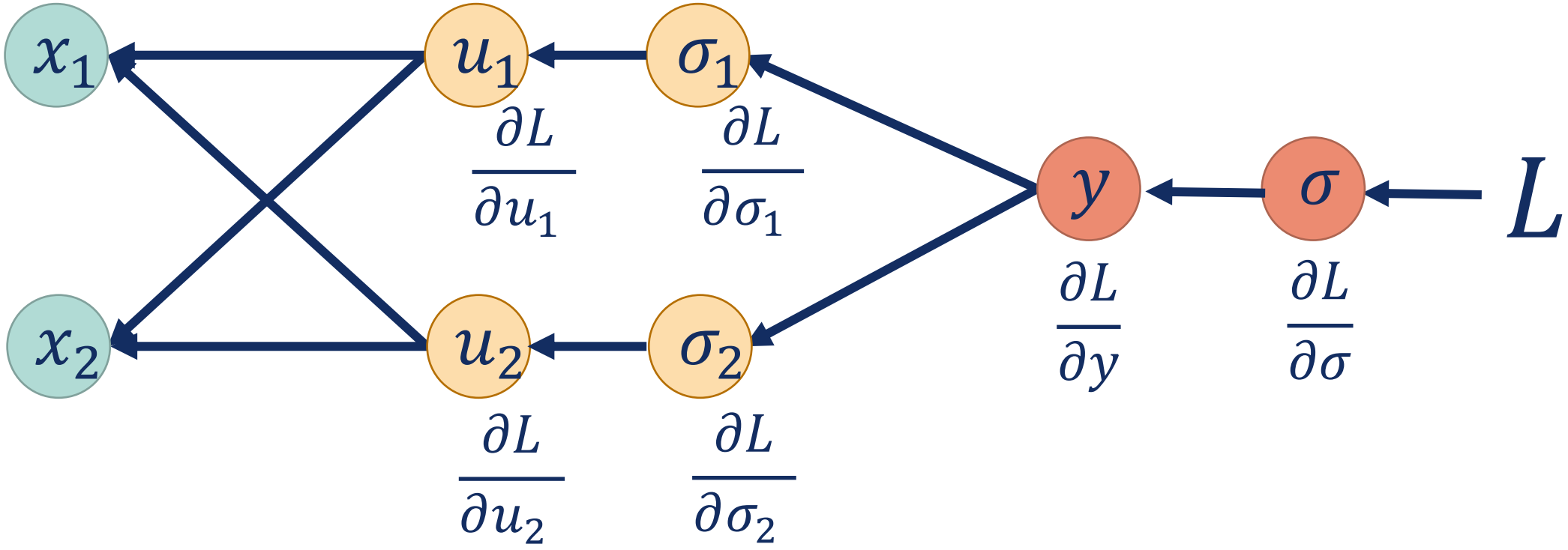
Чередуем **forward pass** (вычисление значений в нейронах)



Этот шаг нам нужен, чтобы знать, в каких точках считать производные

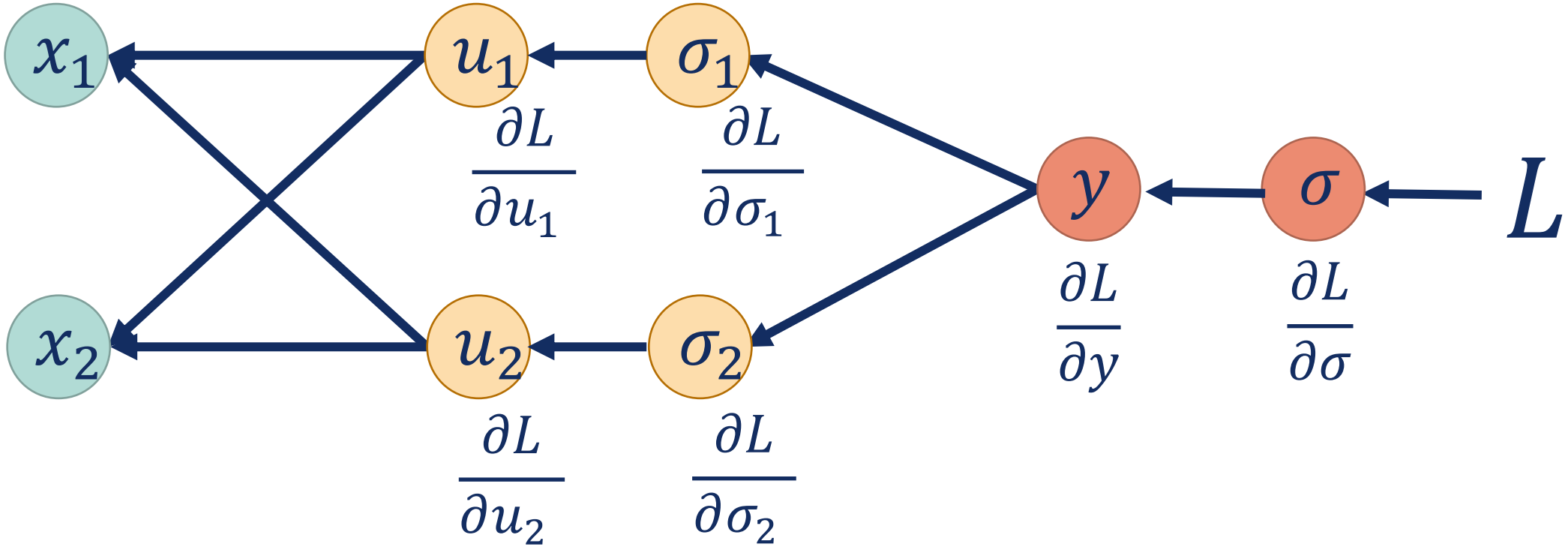
Backprop: как делать

И **backward pass** (вычисление производных):



Backprop: как делать

И **backward pass** (вычисление производных):



Код: https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/
Формулы (стр. 13): http://www.ccas.ru/voron/download/NeuralNets.pdf

Как еще можно понять backprop

- Если вам ближе код, см. реализацию на Python: https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/
- Если вам ближе формулы, см. стр. 13 в лекциях К.В. Воронцова по нейросетям: http://www.ccas.ru/voron/download/NeuralNets.pdf
- Если нужно неторопливое и подробное объяснение, см. стр. 200-217 в Deep Learning Book: https://www.deeplearningbook.org/contents/mlp.html

Как это реализовано в библиотеках

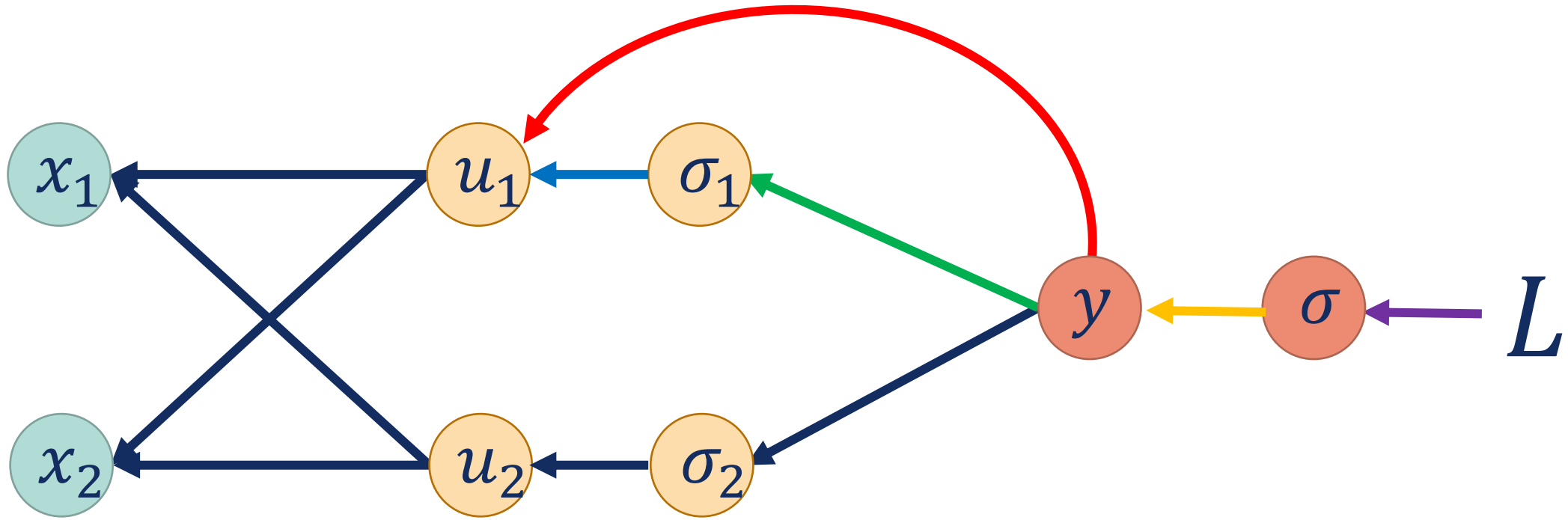
1. Для каждого типа слоя написан forward pass и backward pass
2. Операции оптимизированы за счет матричной записи и алгоритмов быстрых матричных вычислений (см. BLAS)

Трудности обучения нейросетей

1. Взрыв и затухание градиента
2. Все проблемы SGD, в частности – застревание в локальных минимумах и легкое переобучение

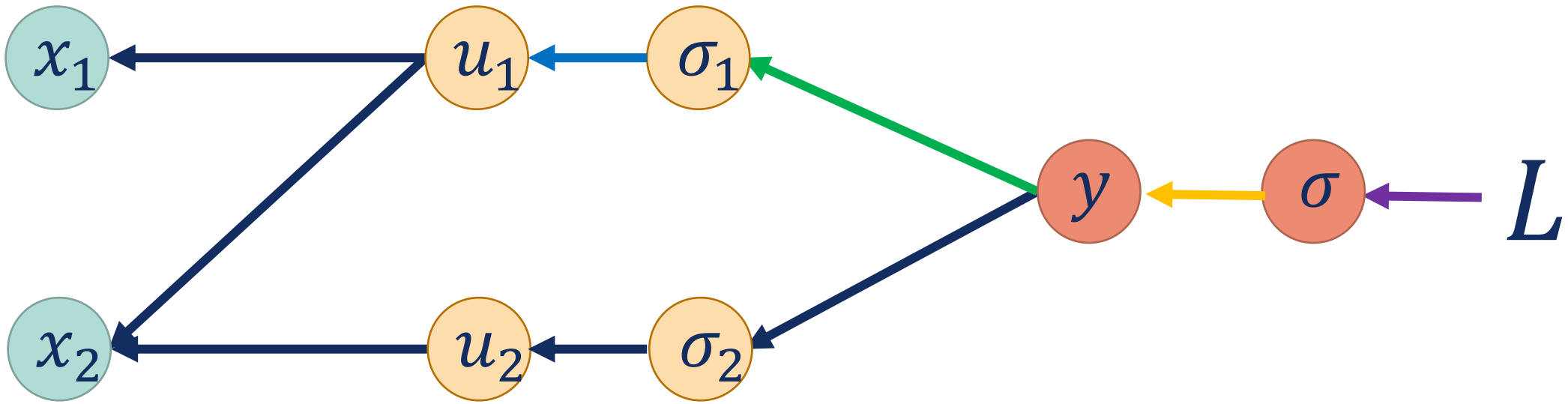
Пример: как архитектура помогает обучению

Добавляем «перепрыгивающие» через слои связи *(residual connection)*: теперь градиент будет «дотекать» не затухая, а большие градиенты просто будем обрезать *(gradient clipping)*

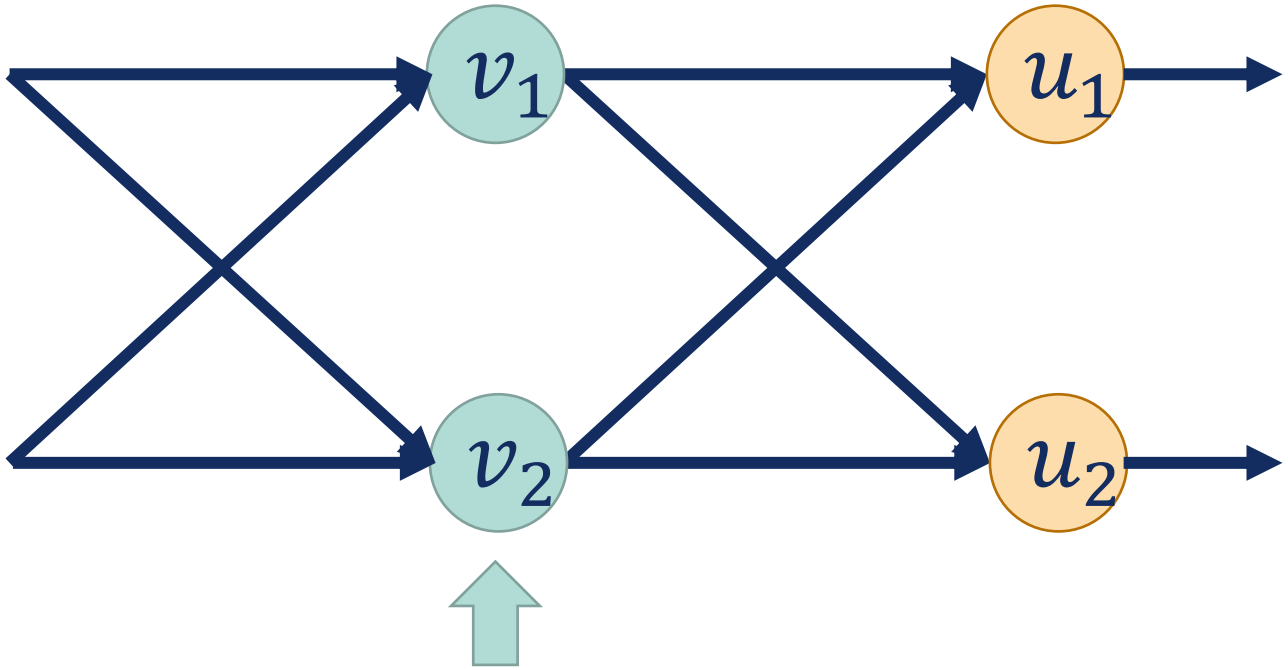


Пример: как архитектура помогает обучению

Другой вариант: сразу убираем часть связей (считаем, что вес на них ноль и не настраиваем его) – меньше параметров, меньше риск переобучиться



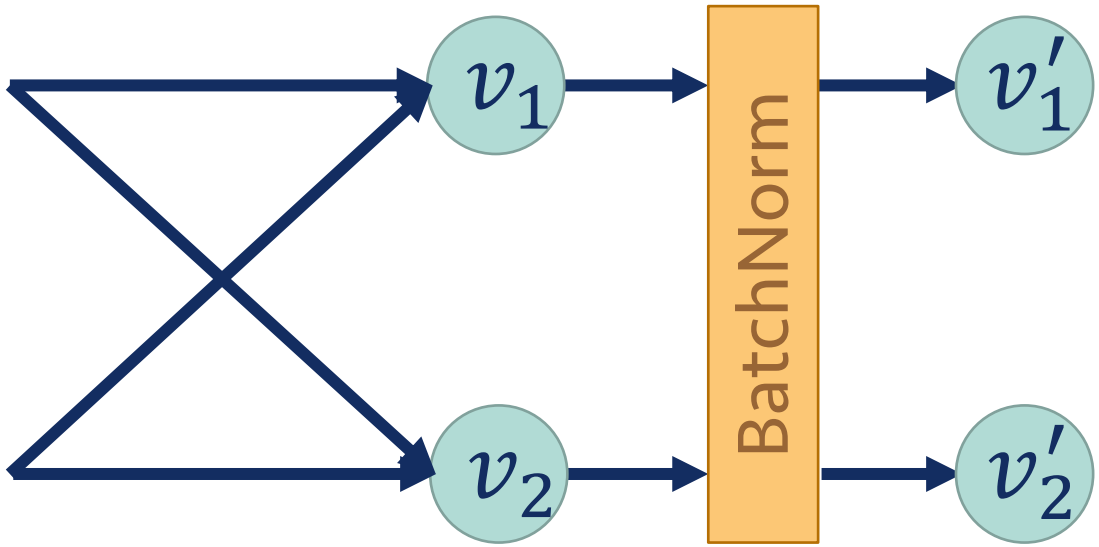
Еще один пример борьбы с переобучением



На прошлой итерации здесь могло быть другое распределение,
т.к. веса на предыдущих слоях тоже поменялись.

Это различие распределений называется *internal covariate shift* (ICS)

Еще один пример борьбы с переобучением



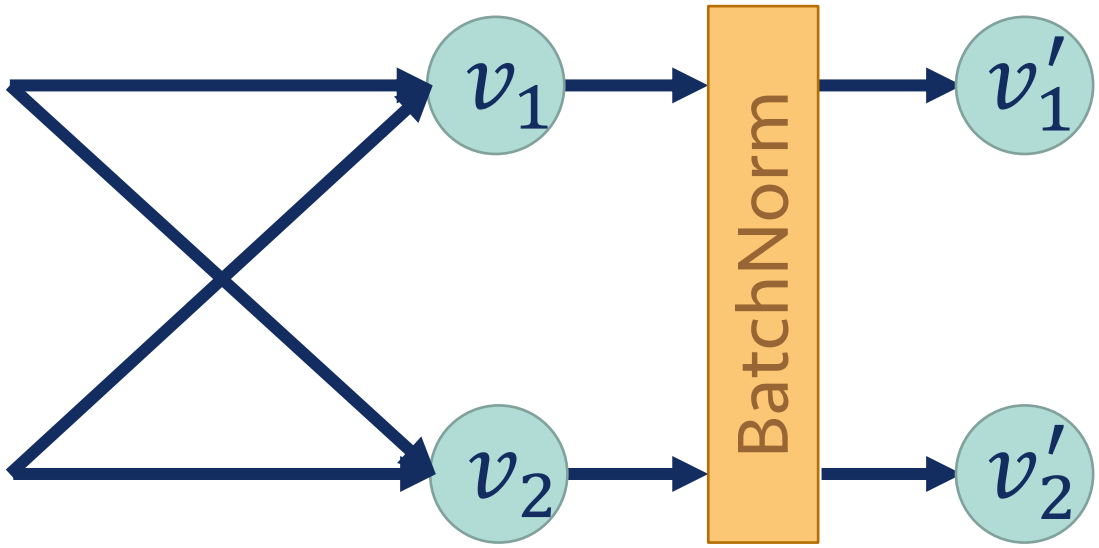
Batch Normalization:
Будем добавлять слои, в которых каждый признак будет нормироваться на среднее и дисперсию по батчу

$v_1^{(1)}, \dots, v_1^{(m)}$ – значения признака v_1 по батчу размера m

Среднее по батчу: $\mu = \frac{1}{m}\sum_{i=1}^{m} v_1^{(i)}$

Дисперсия по батчу: $\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}\left(v_1^{(i)} - \mu\right)^2$

Еще один пример борьбы с переобучением

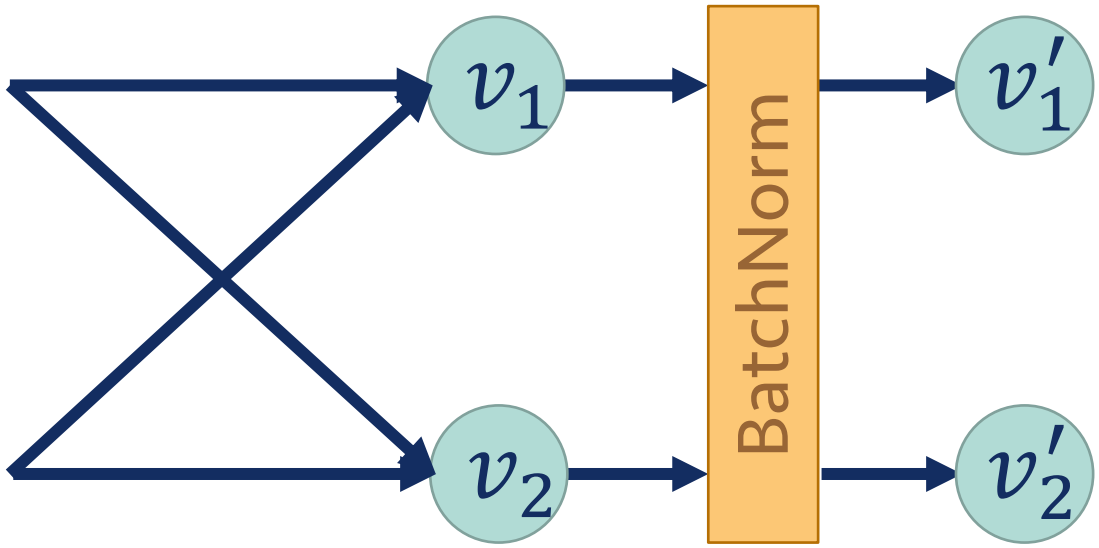


Batch Normalization:
Будем добавлять слои, в которых каждый признак будет нормироваться на среднее и дисперсию по батчу

Нормировка: $\tilde{v}_1 = \frac{v_1 - \mu}{\sqrt{\sigma^2 + \varepsilon}}$

Масштабирование и сдвиг: $v_1' = \gamma \tilde{v}_1 + \beta$

Еще один пример борьбы с переобучением

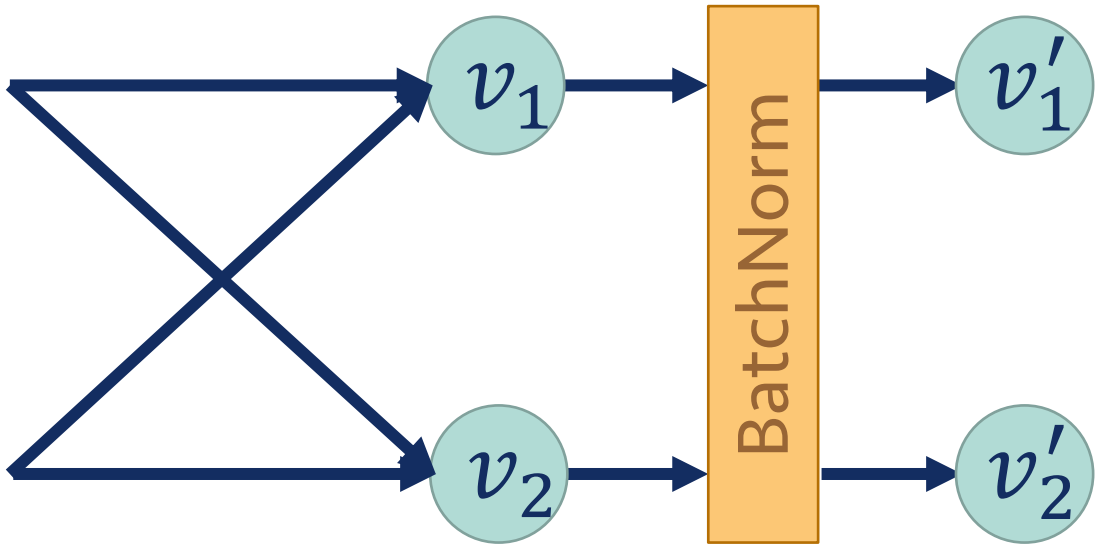


Batch Normalization:
Будем добавлять слои, в которых каждый признак будет нормироваться на среднее и дисперсию по батчу

Нормировка: $\tilde{v}_1 = \frac{v_1 - \mu}{\sqrt{\sigma^2 + \varepsilon}}$

Масштабирование и сдвиг: $v_1' = \gamma \tilde{v}_1 + \beta$

Еще один пример борьбы с переобучением



Batch Normalization:
Будем добавлять слои, в которых каждый признак будет нормироваться на среднее и дисперсию по батчу

Нормировка: $\tilde{v}_1 = \frac{v_1 - \mu}{\sqrt{\sigma^2 + \varepsilon}}$

Масштабирование и сдвиг: $v_1' = \gamma \tilde{v}_1 + \beta$

Полезный вопрос: зачем второй шаг?

Немного драмы: ICS не причём (NIPS 2018)

How Does Batch Normalization Help Optimization?

Shibani Santurkar* MIT shibani@mit.edu
Dimitris Tsipras* MIT tsipras@mit.edu
Andrew Ilyas* MIT ailyas@mit.edu
Aleksander Mądry MIT madry@mit.edu

Abstract

Batch Normalization (BatchNorm) is a widely adopted technique that enables faster and more stable training of deep neural networks (DNNs). Despite its pervasiveness, the exact reasons for BatchNorm's effectiveness are still poorly understood. The popular belief is that this effectiveness stems from controlling the change of the layers' input distributions during training to reduce the so-called "internal covariate shift". In this work, we demonstrate that such distributional stability of layer inputs has little to do with the success of BatchNorm. Instead, we uncover a more fundamental impact of BatchNorm on the training process: it makes the optimization landscape significantly smoother. This smoothness induces a more predictive and stable behavior of the gradients, allowing for faster training.

https://arxiv.org/abs/1805.11604

2. Рекуррентные нейросети

Рекуррентная нейросеть как формула

$x_1, \ldots, x_l$ — векторные представления последовательно идущих слов из текста

Рекуррентная нейросеть как формула

$x_1, \ldots, x_l$ — векторные представления последовательно идущих слов из текста

Скрытое представление текста к i-тому слову:

$$h_i = f_h(W_{xh}x_i + W_{hh}h_{i-1} + b_h)$$

Рекуррентная нейросеть как формула

$x_1, \ldots, x_l$ — векторные представления последовательно идущих слов из текста

Скрытое представление текста к i-тому слову:

$$h_i = f_h(W_{xh}x_i + W_{hh}h_{i-1} + b_h)$$

Прогноз ответа по тексту от 1 до i-того слова:

$$\hat{y}_i = f_y\left(W_{hy}h_i + b_y\right)$$

Рекуррентная нейросеть как формула

Значения скрытого слоя на каждом слове:

$$h_1 = f_h(W_{xh}x_1 + 0 + b_h)$$
$$h_2 = f_h(W_{xh}x_2 + W_{hh}h_1 + b_h)$$
$$h_3 = f_h(W_{xh}x_3 + W_{hh}h_2 + b_h)$$
$$\dots$$

Схема RNN

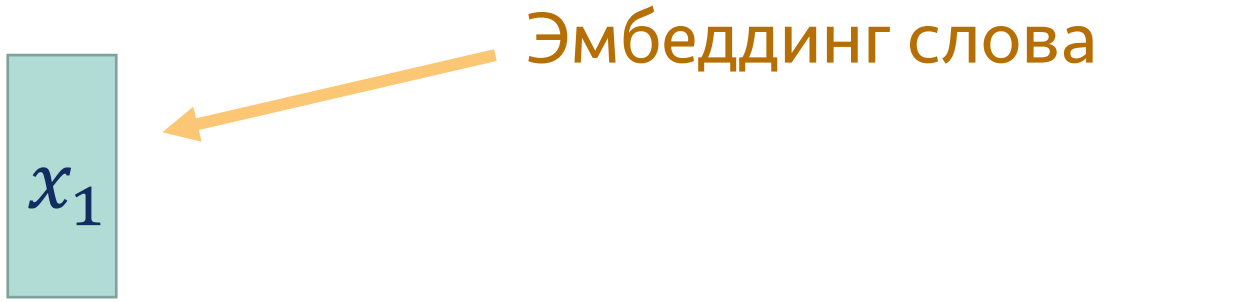
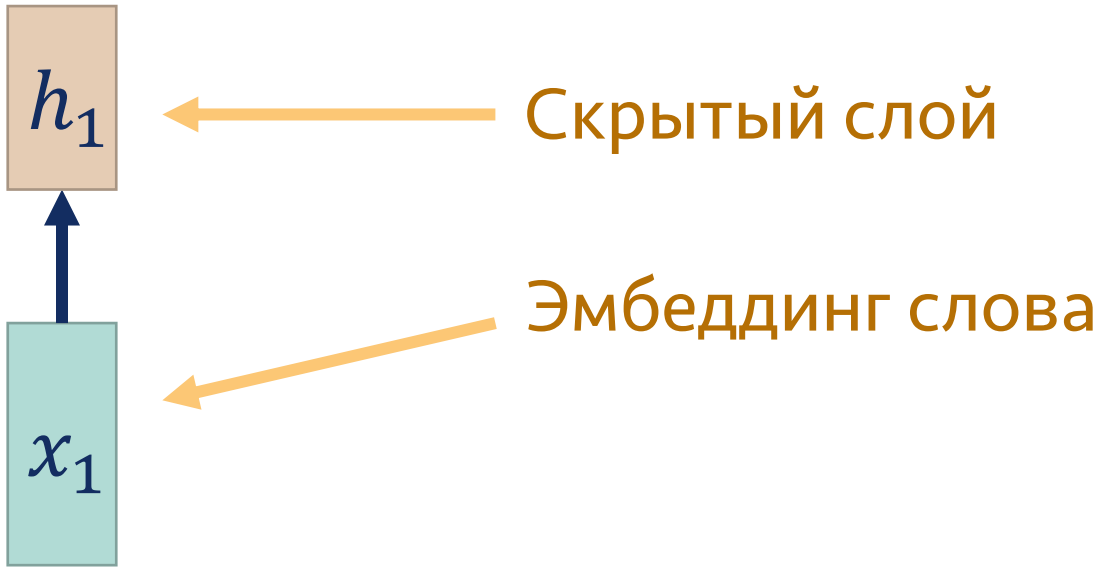
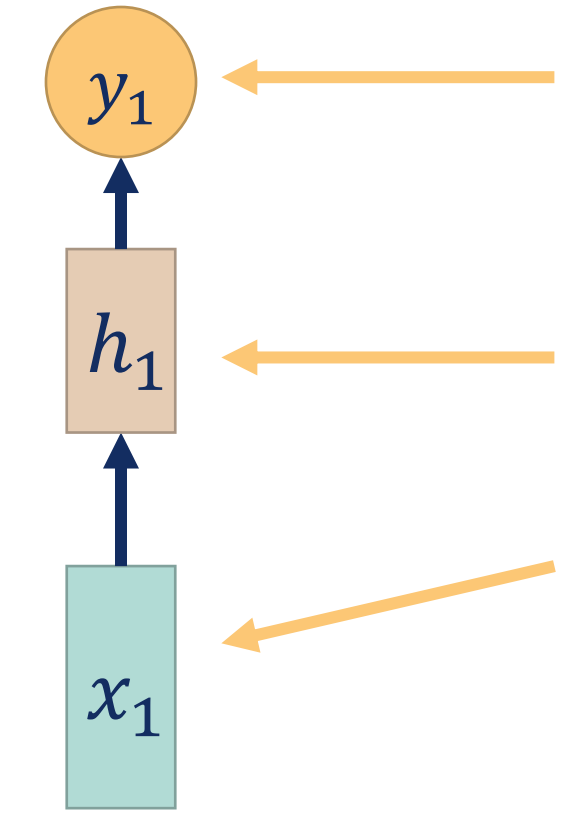


Схема RNN



Вчера телефон перестал работать

Схема RNN



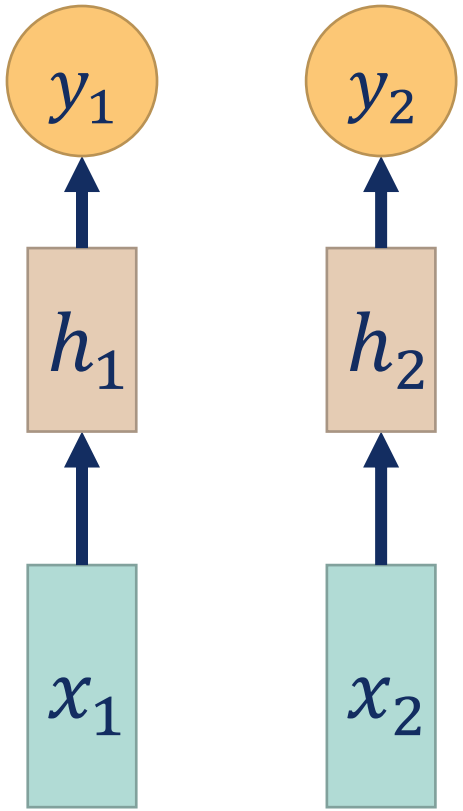
Ответ (например, вероятность того, что отзыв позитивный)

Скрытый слой

Эмбеддинг слова

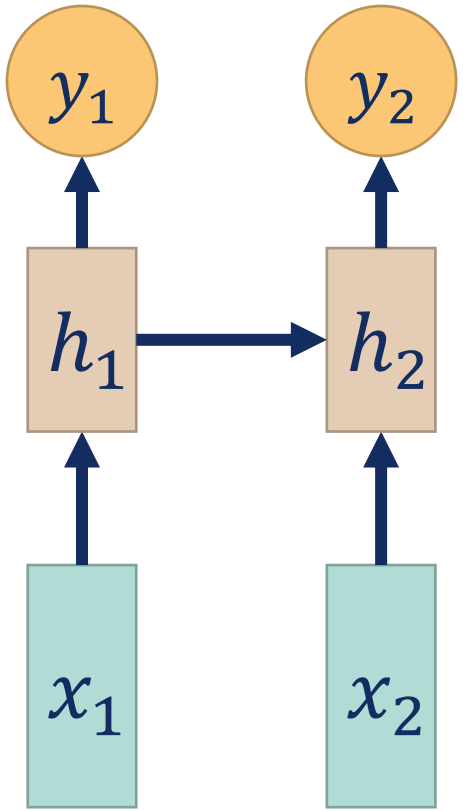
Вчера телефон перестал работать

Схема RNN



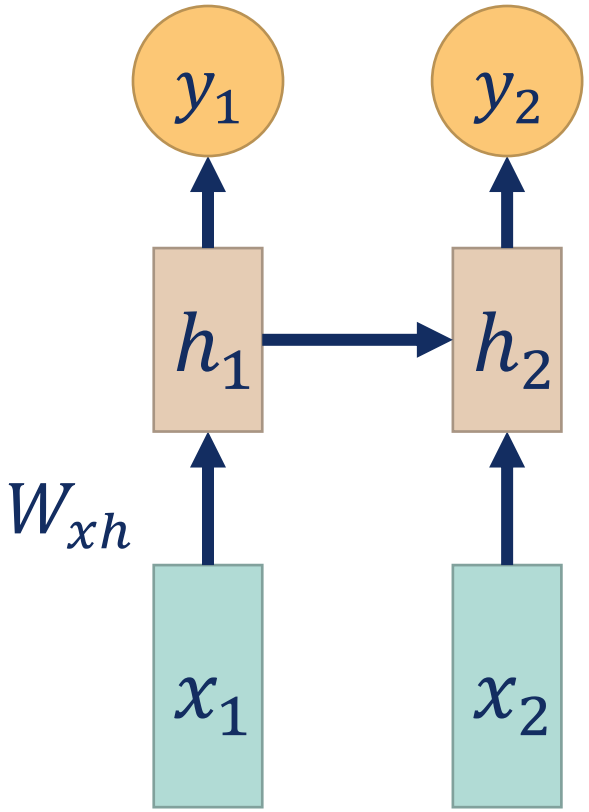
Вчера телефон перестал работать

Схема RNN



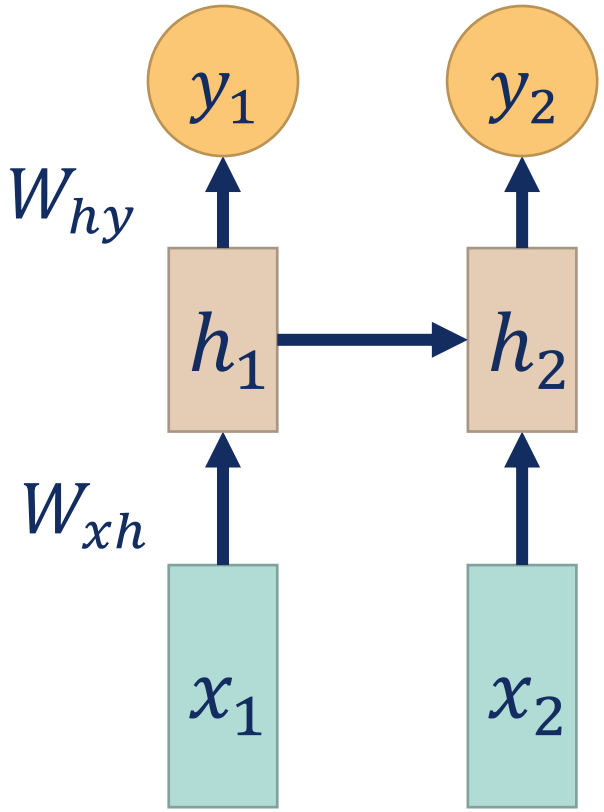
Вчера телефон перестал работать

Схема RNN



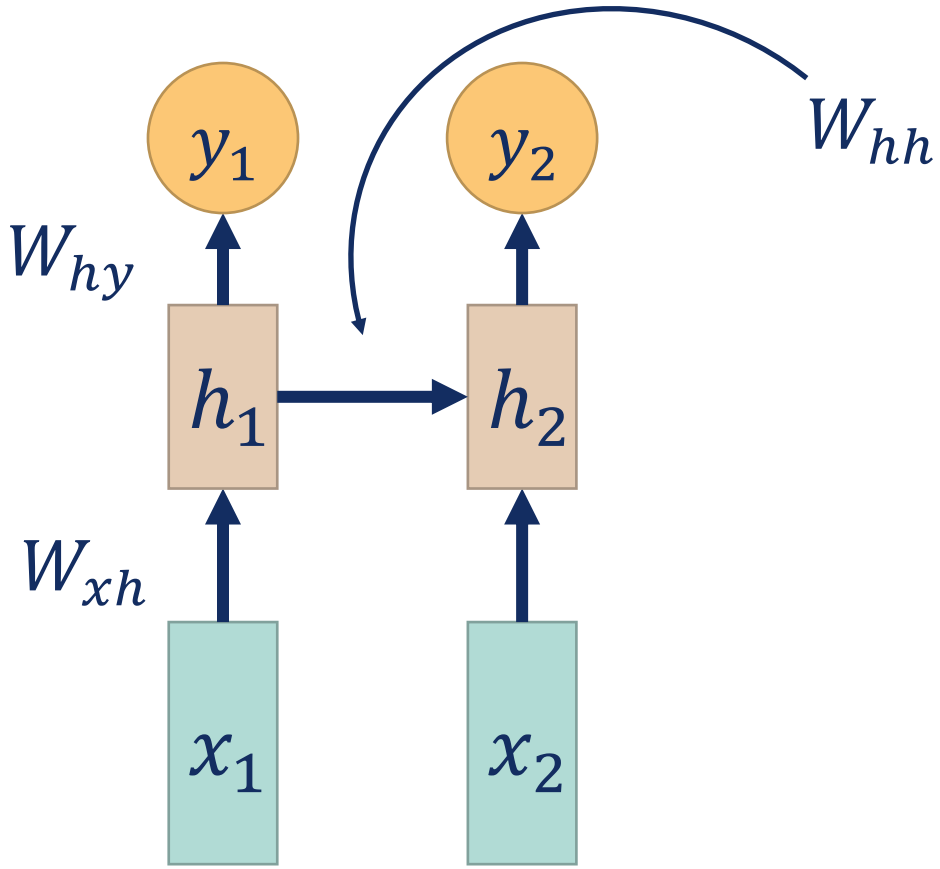
Вчера телефон перестал работать

Схема RNN



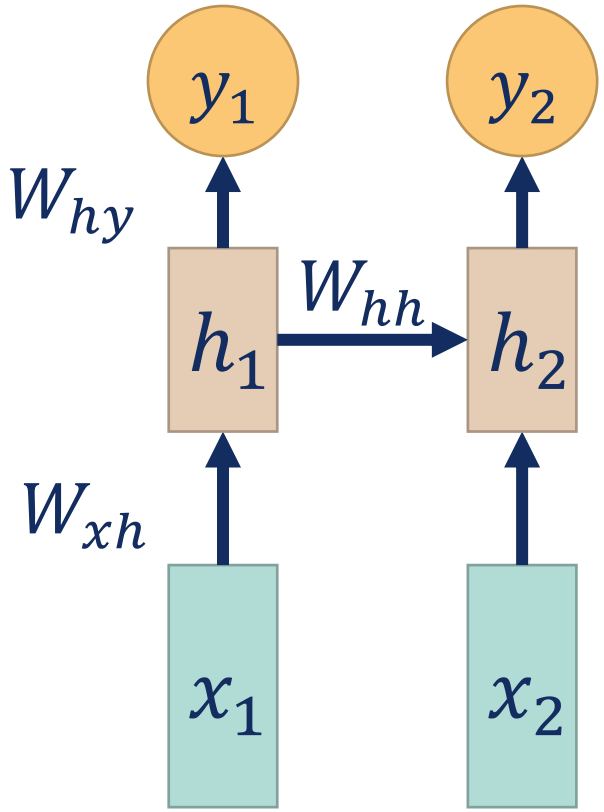
Вчера телефон перестал работать

Схема RNN



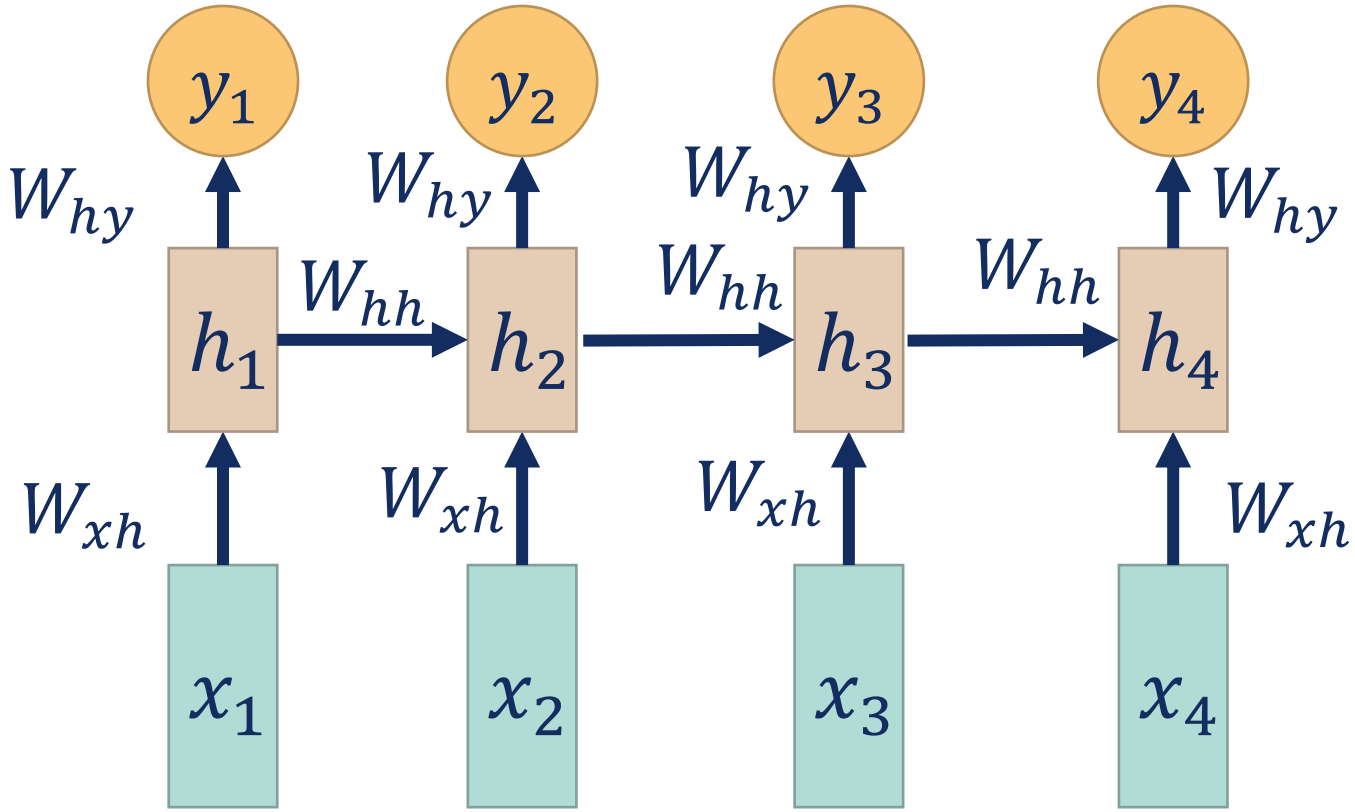
Вчера телефон перестал работать

Схема RNN



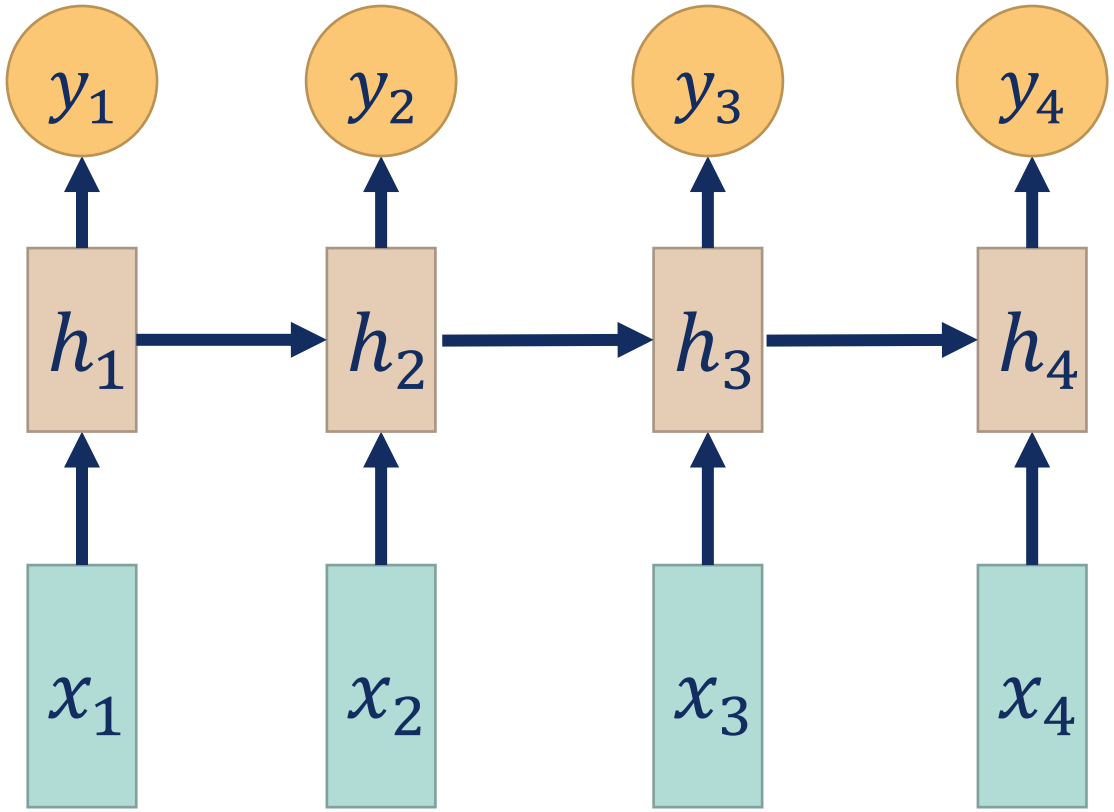
Вчера телефон перестал работать

Схема RNN



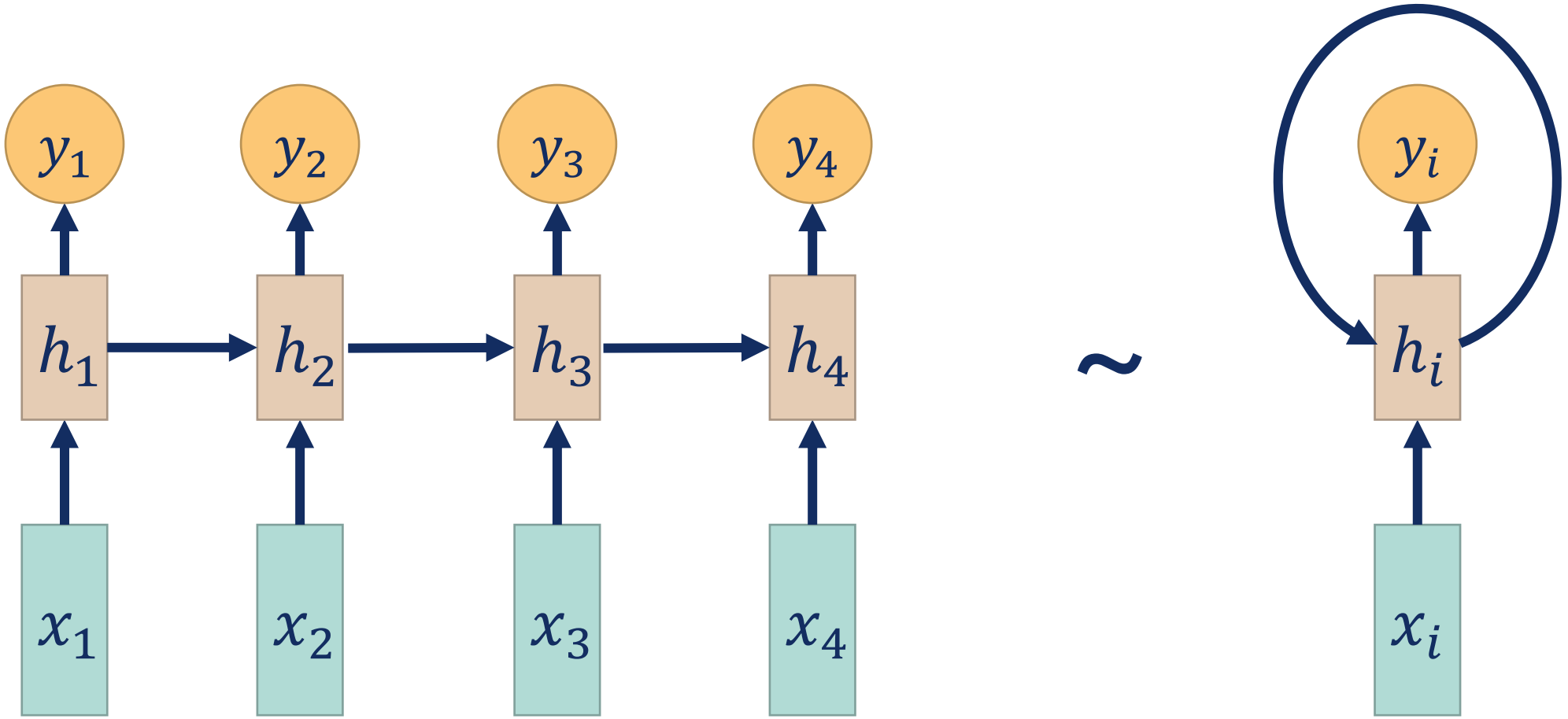
Вчера телефон перестал работать

Схема RNN



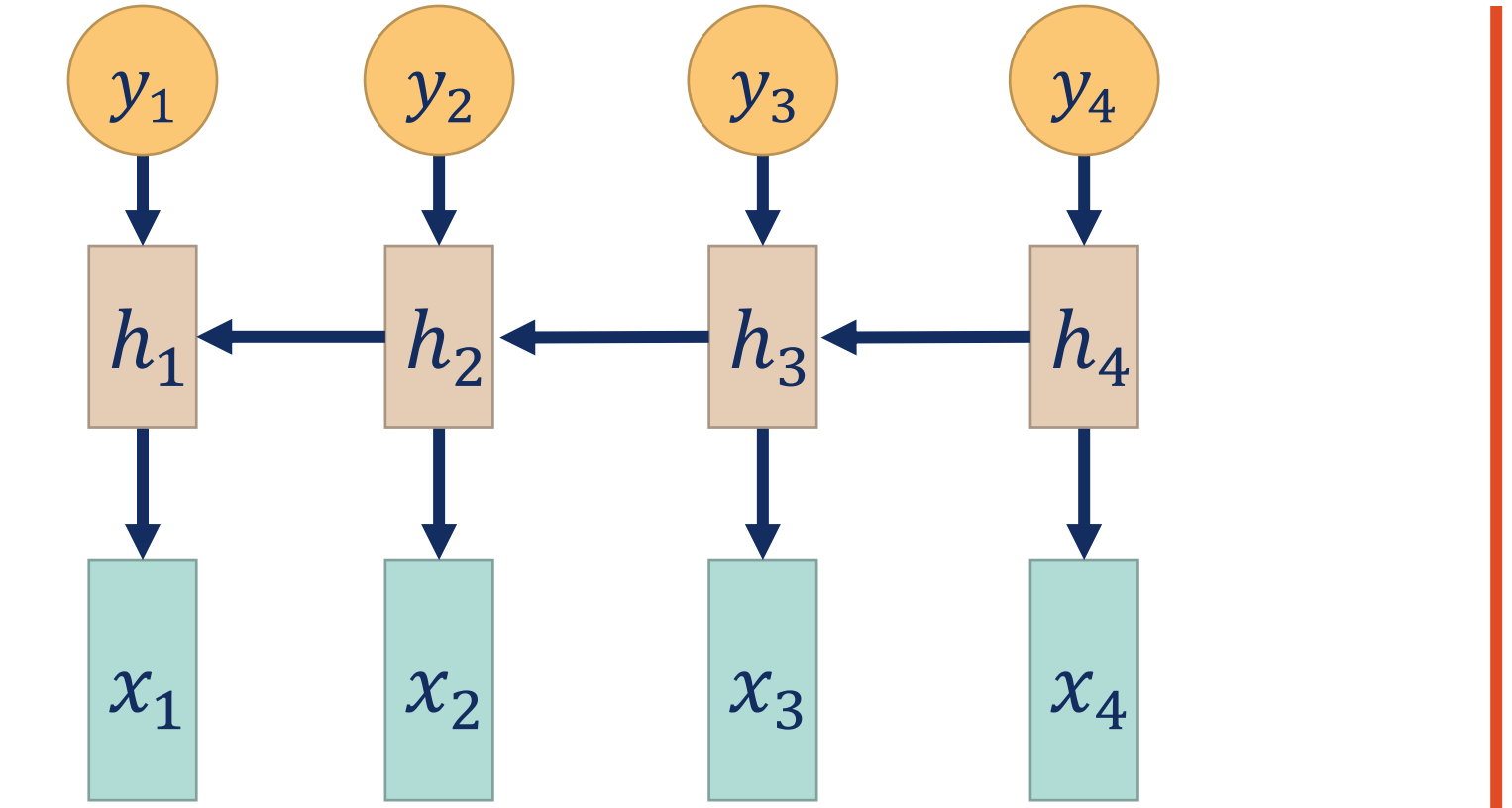
Вчера телефон перестал работать

Схема RNN



Вчера телефон перестал работать

Обучение RNN: backpropagation through time



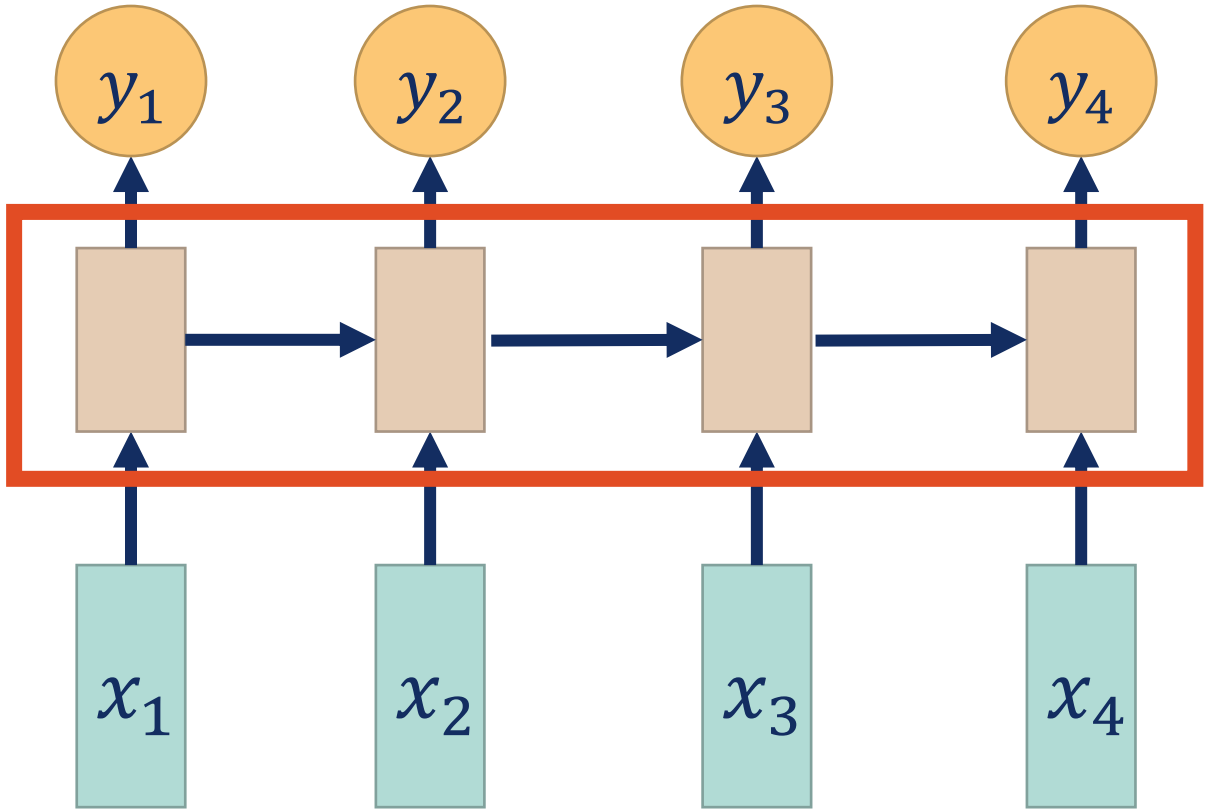
Основная проблема Vanilla RNN

Учитывать длинные последовательности слов сложно из-за затухания градиентов

Вся следующая секция посвящена борьбе с этой проблемой

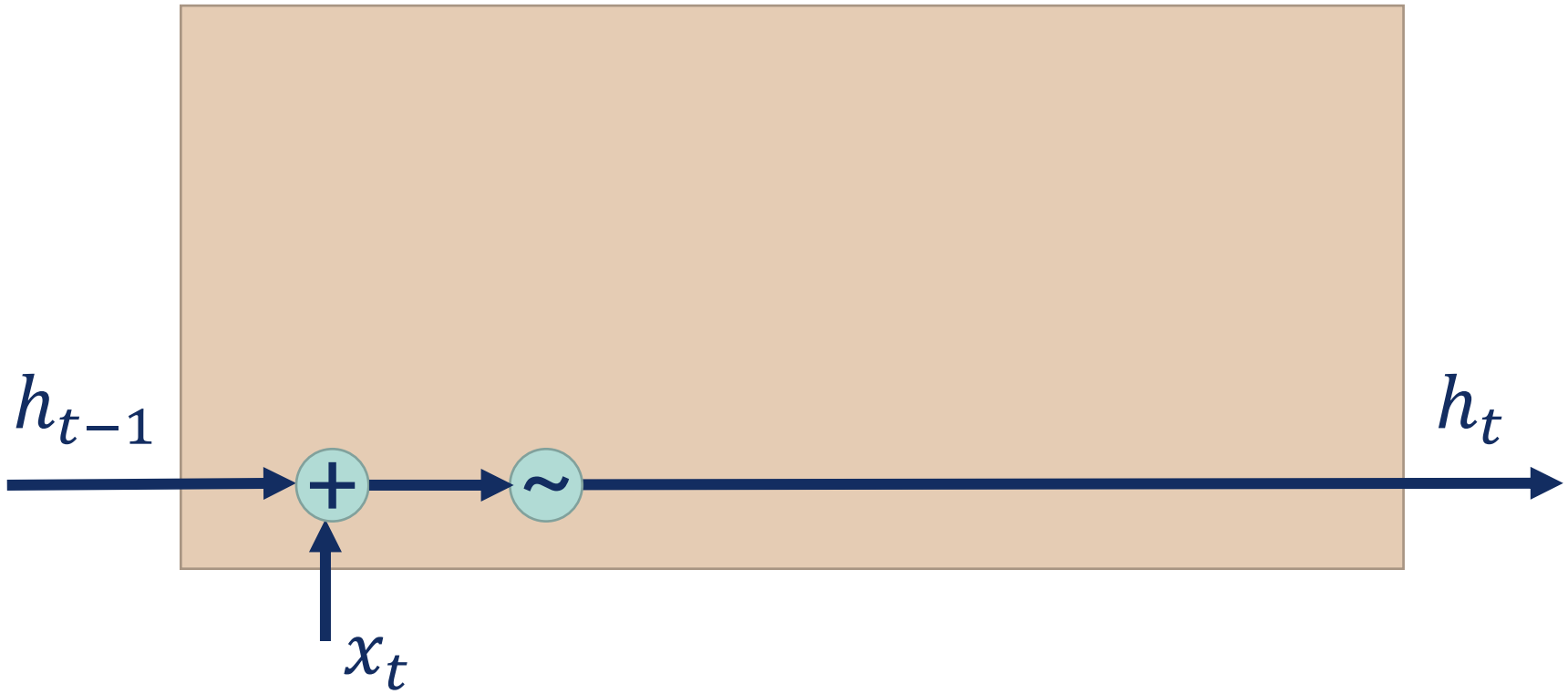
3. LSTM и GRU

Что будем модифицировать в этой секции



Вчера телефон перестал работать

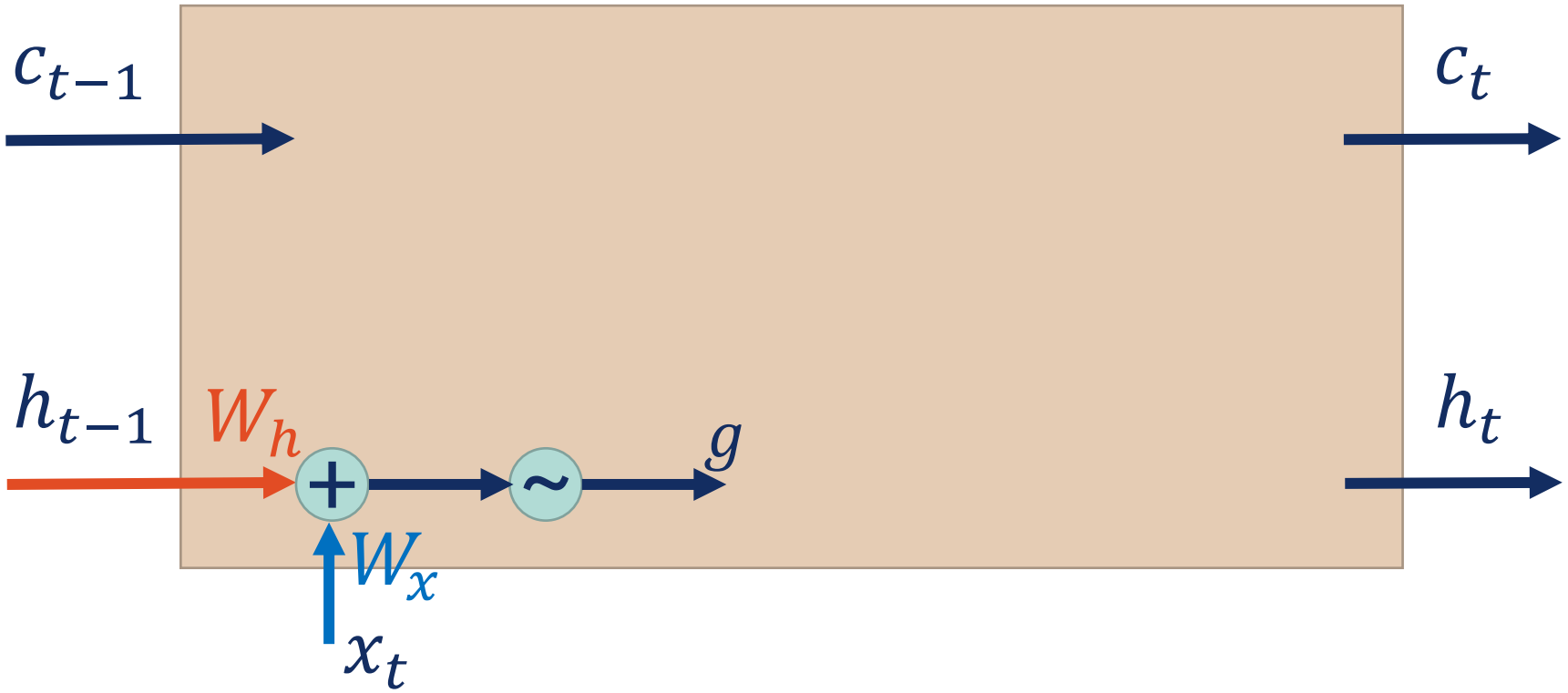
Как было в RNN



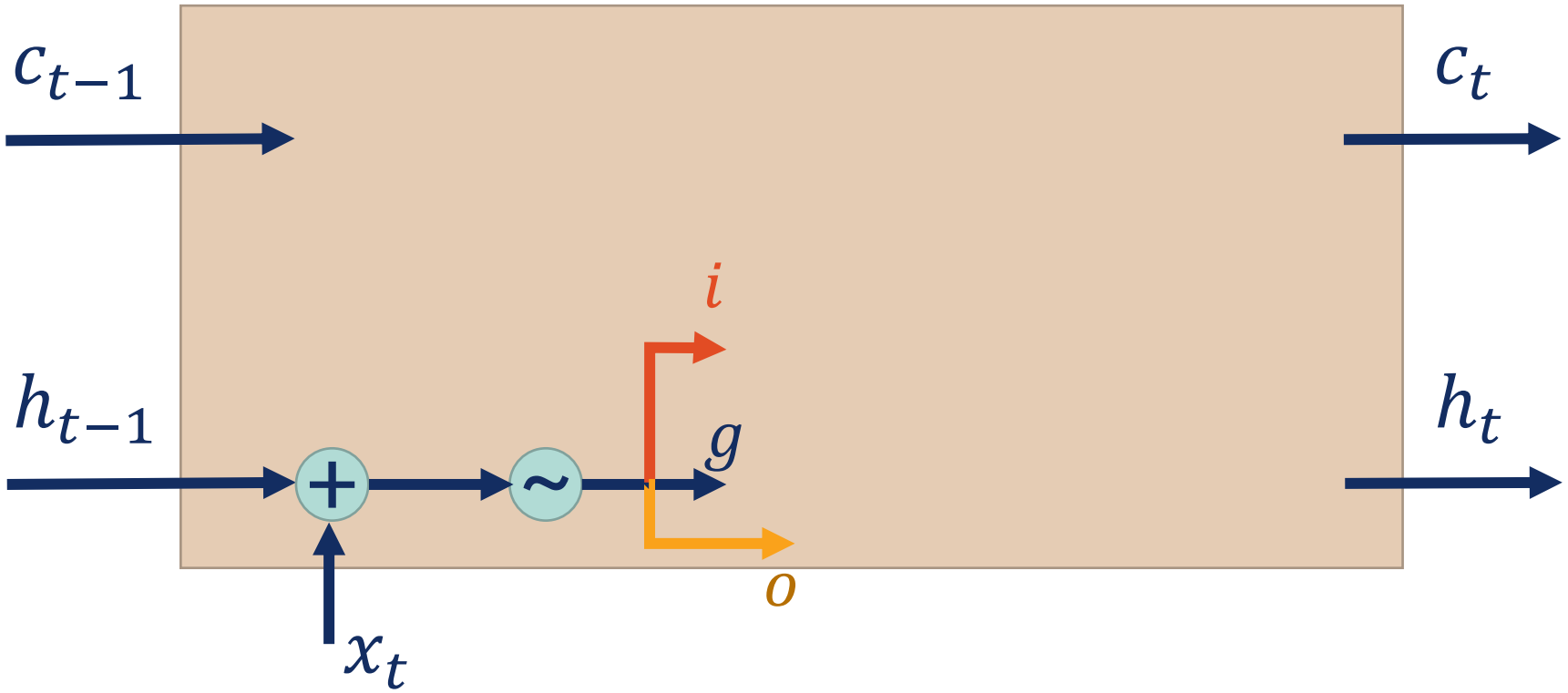
Добавляем память

Добавляем память

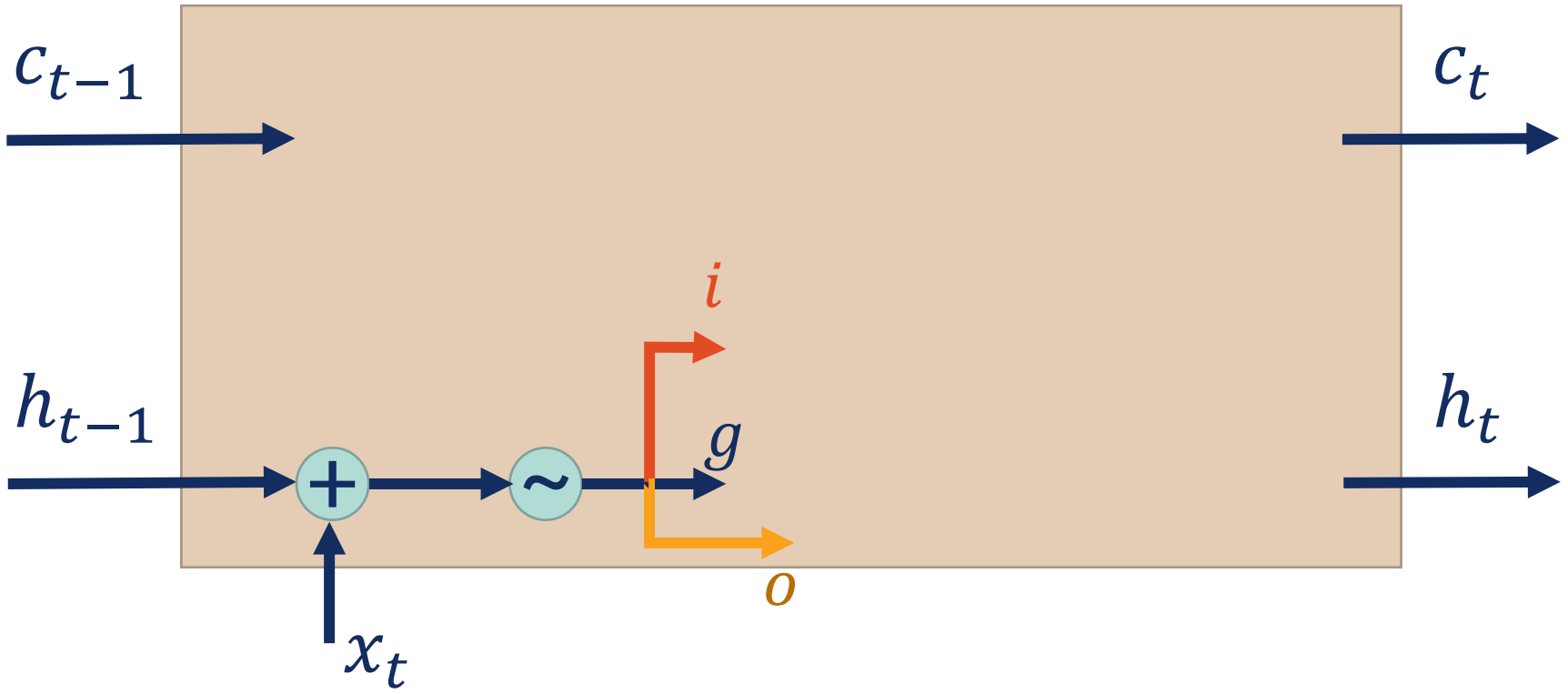
Добавляем память



Добавляем память

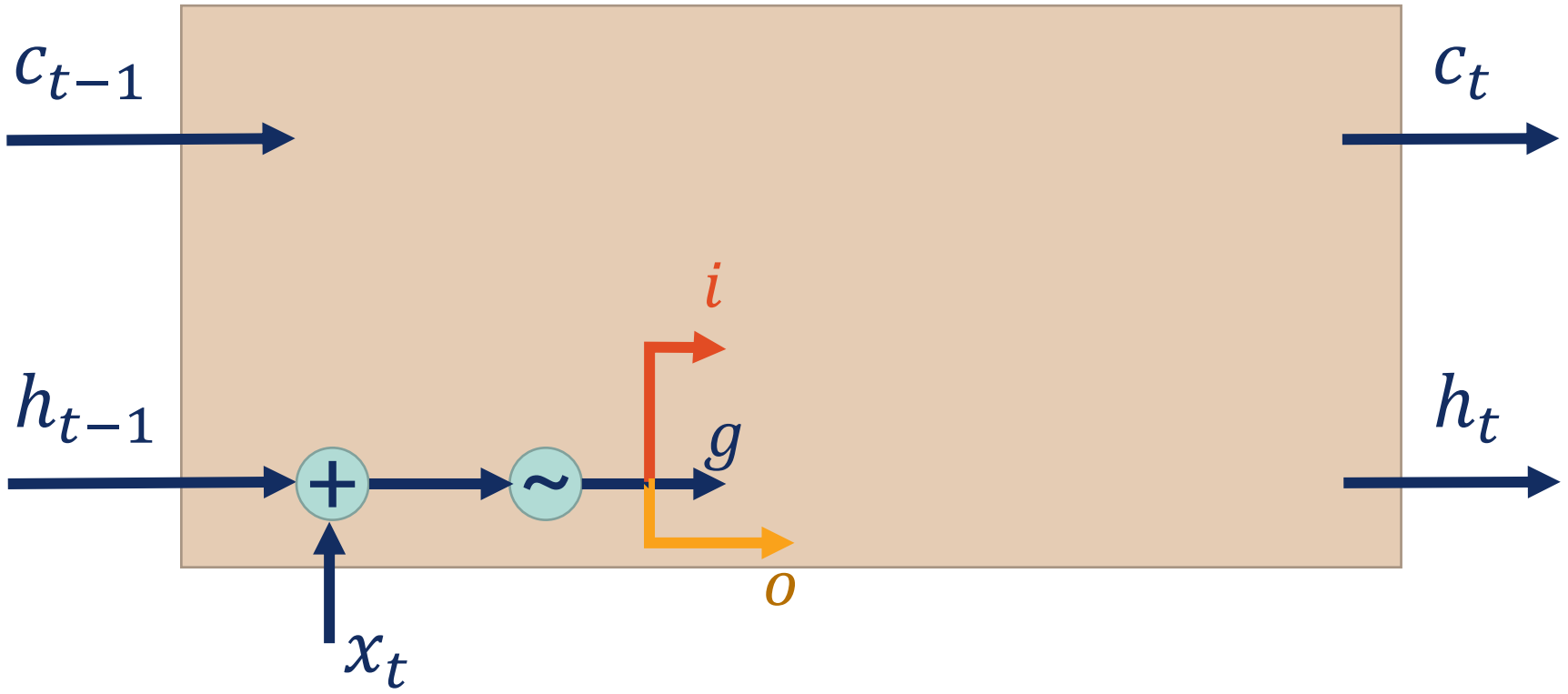


Добавляем память



$$g_t = \varphi\left(W_{xg}x_t + W_{hg}h_{t-1} + b_g\right)$$
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

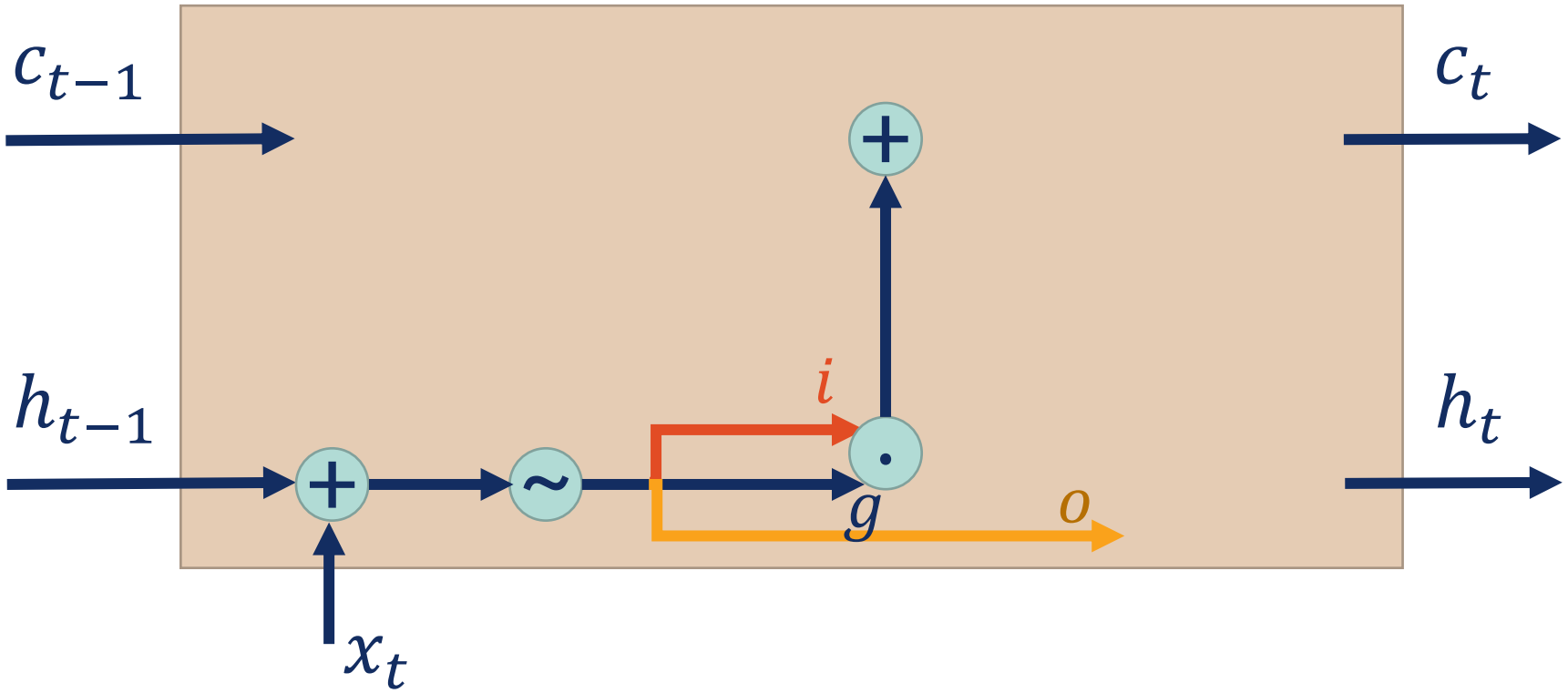
Добавляем память



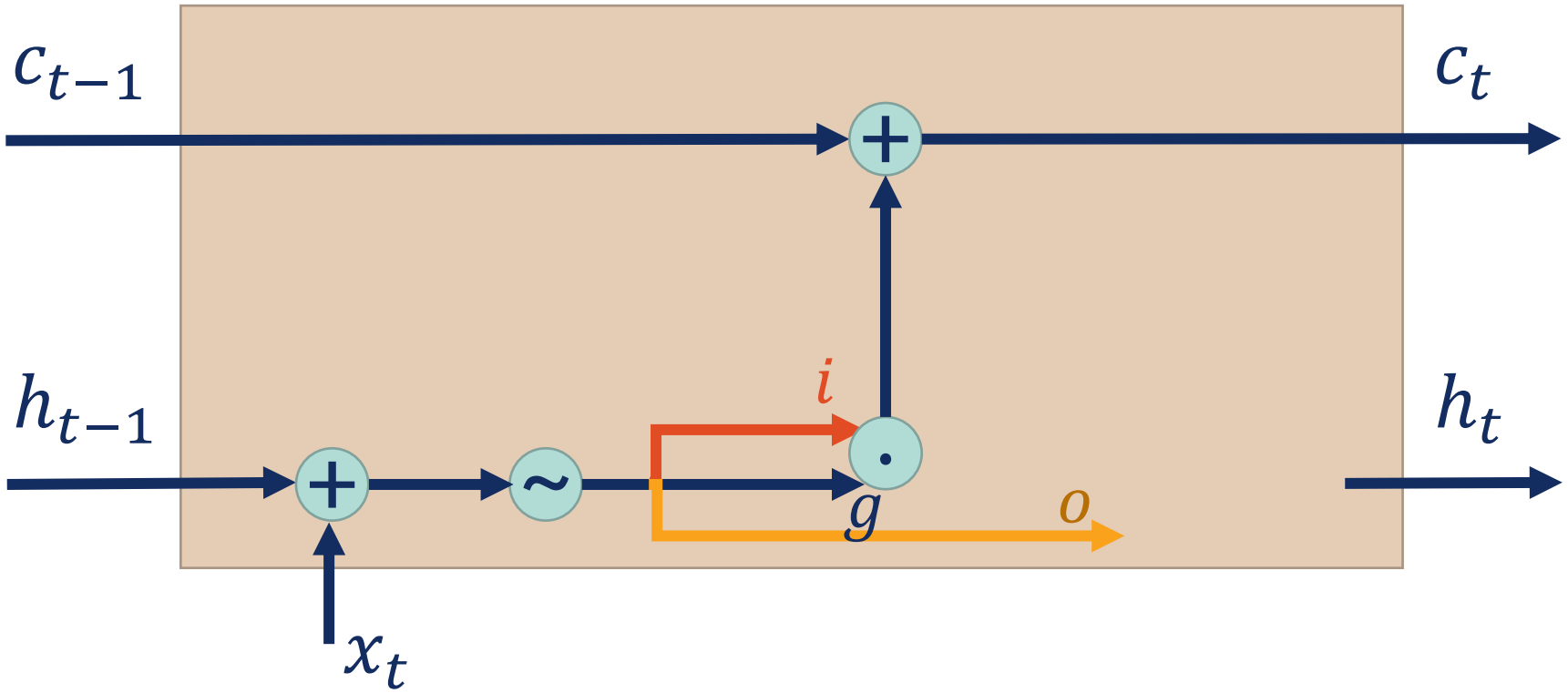
$$g_t = \varphi\left(W_{xg}x_t + W_{hg}h_{t-1} + b_g\right)$$
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

$$\begin{pmatrix} g_t \\ i_t \\ o_t \end{pmatrix} = \begin{pmatrix} \varphi \\ \sigma \\ \sigma \end{pmatrix} (W_x x_t + W_h h_{t-1} + b)$$

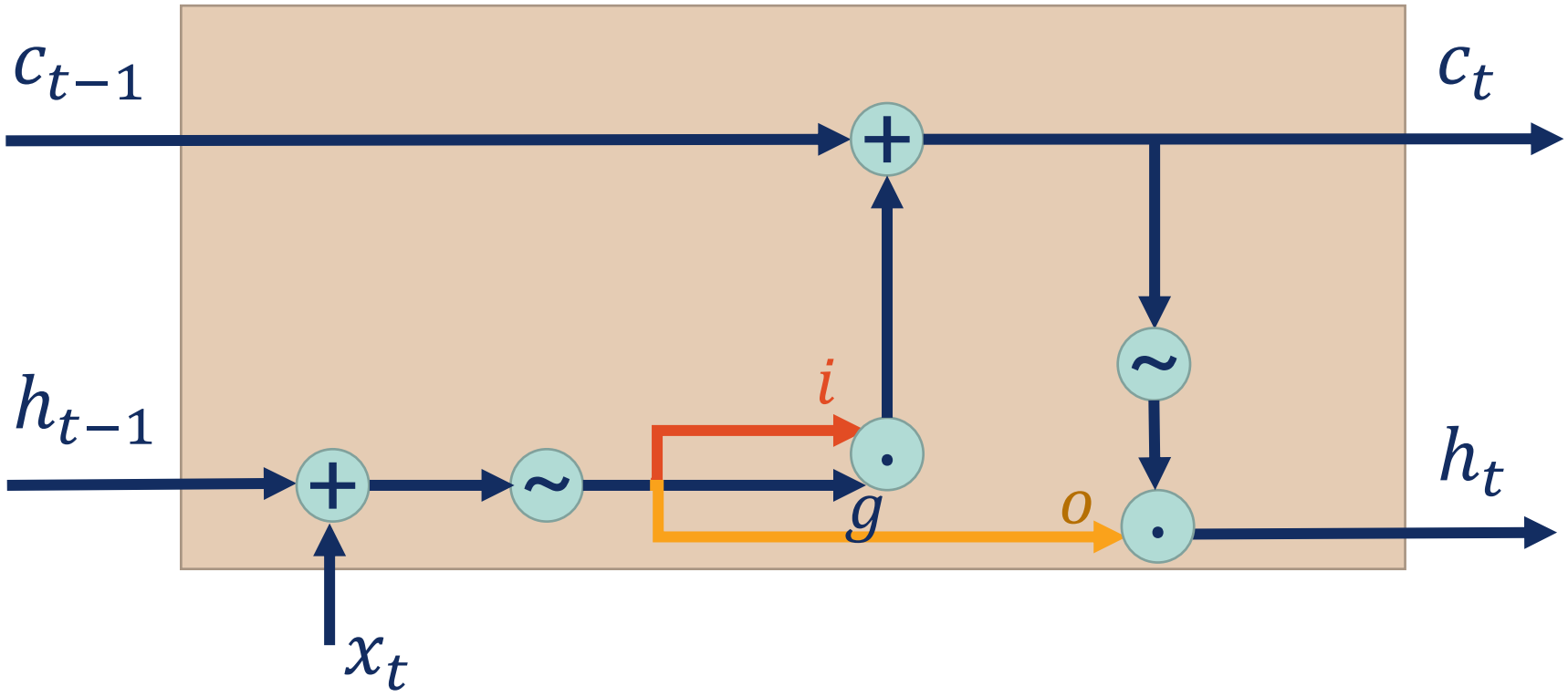
Добавляем память



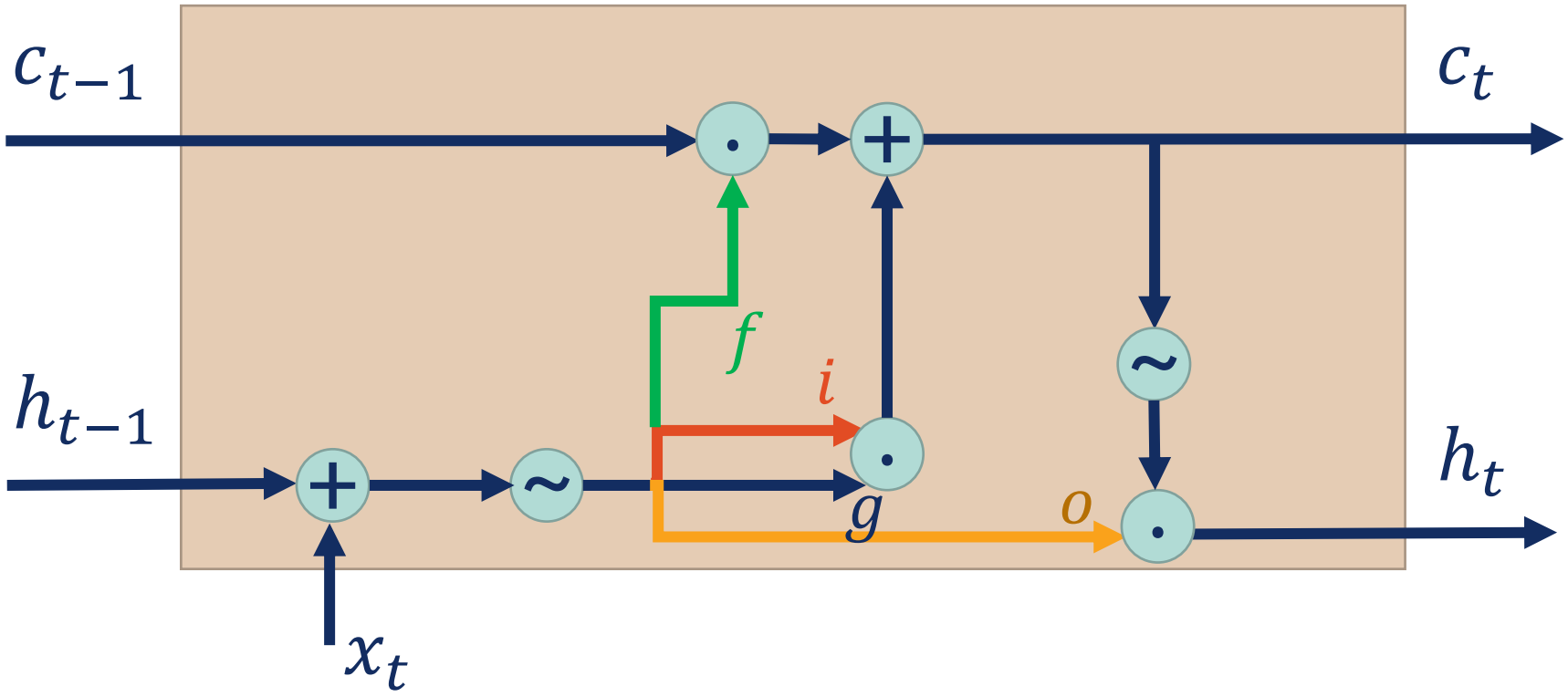
Добавляем память



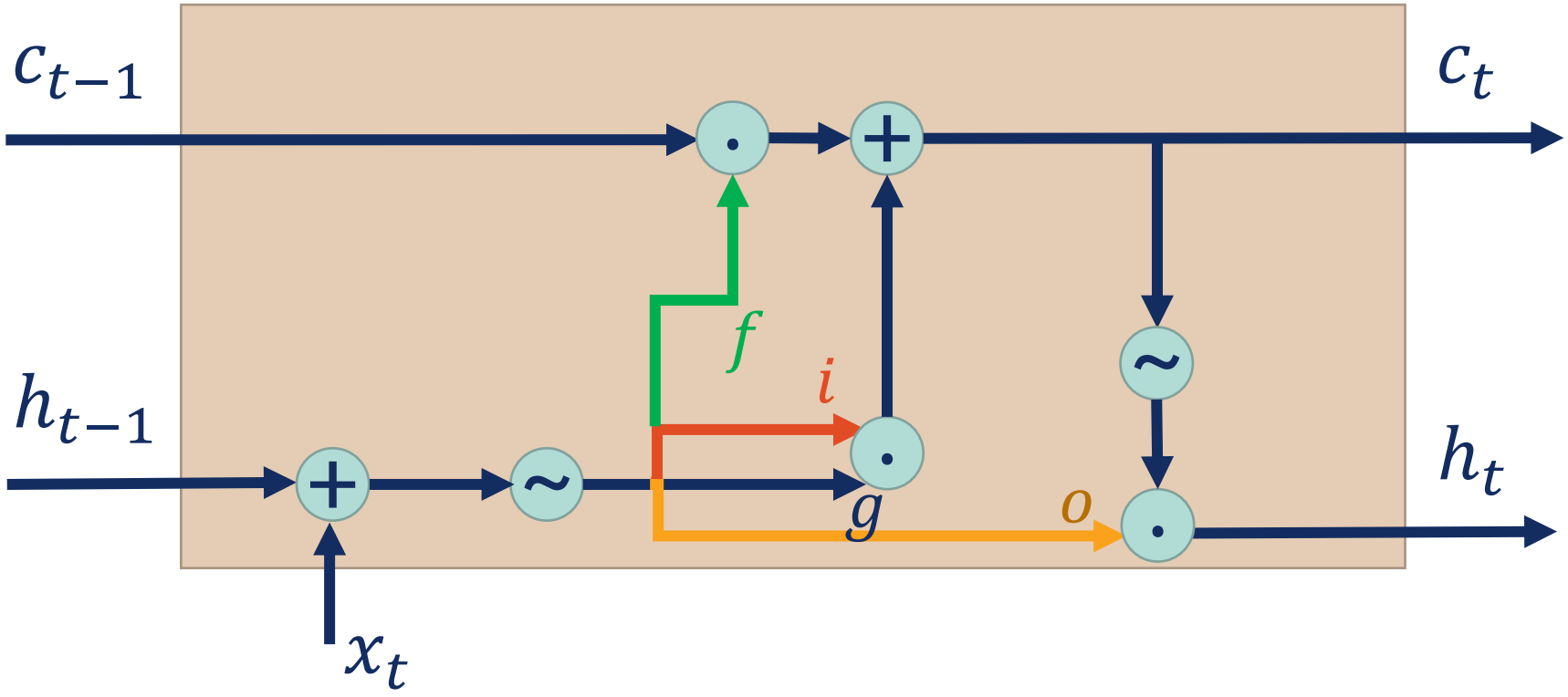
Почти LSTM: не хватает forget gate



Long-Short Term Memory (LSTM)



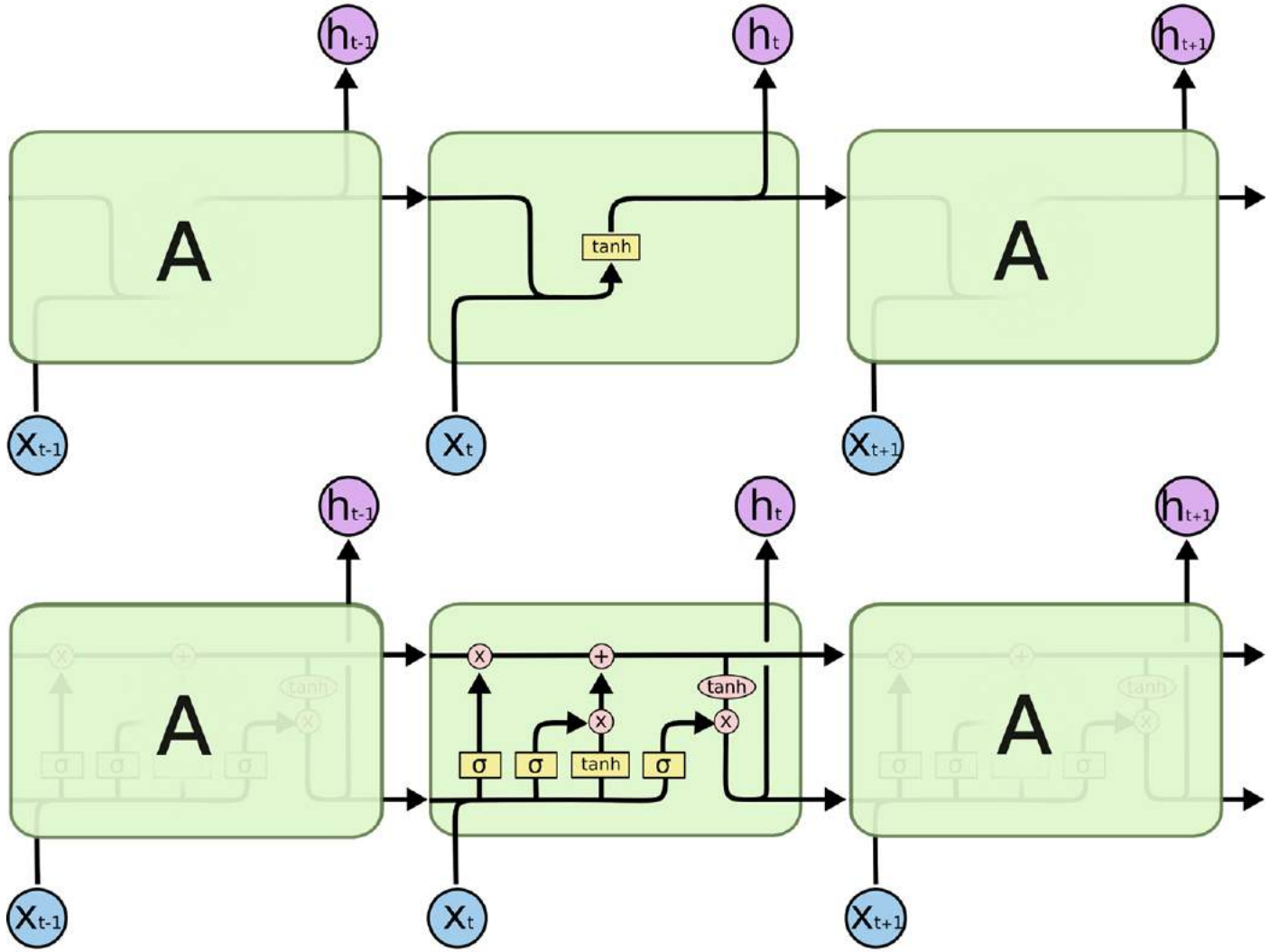
Long-Short Term Memory (LSTM)



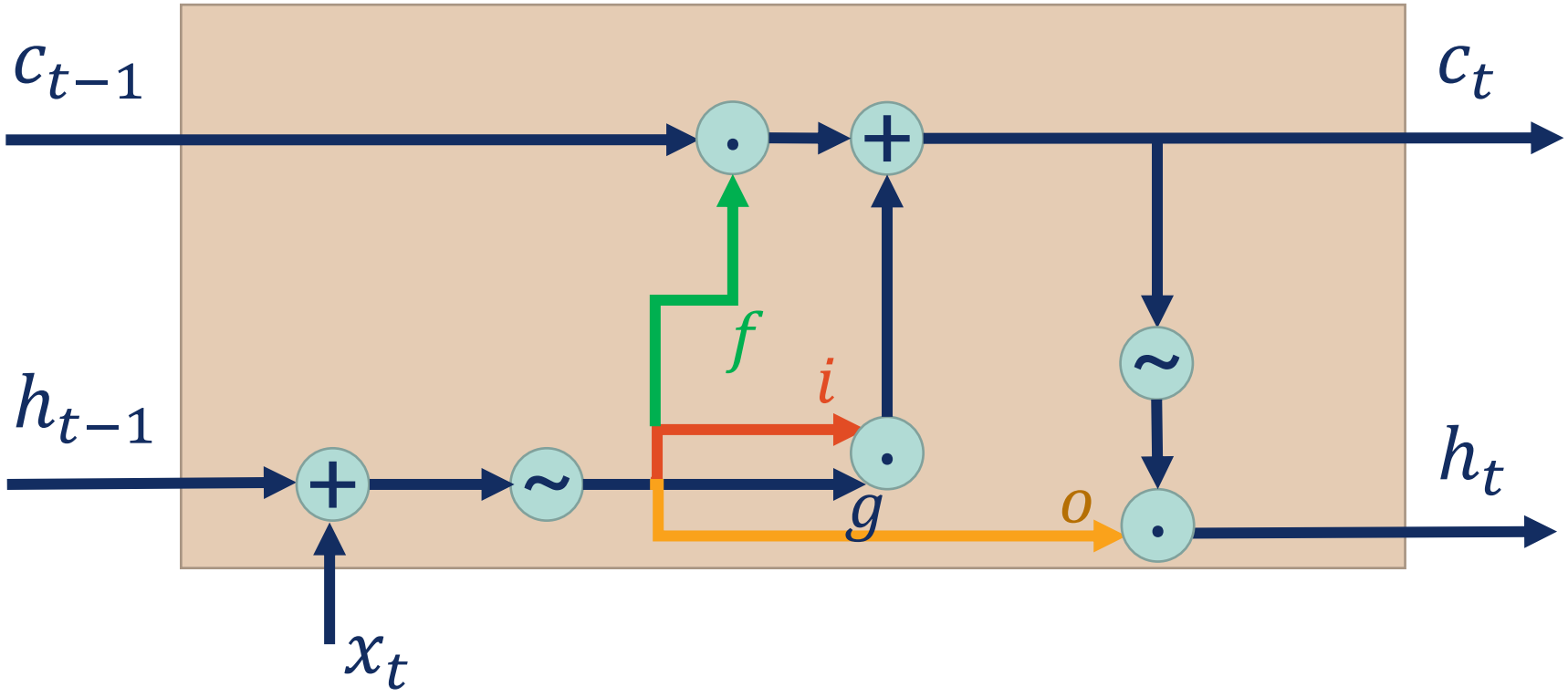
$$\begin{pmatrix} g_t \\ i_t \\ o_t \\ f_t \end{pmatrix} = \begin{pmatrix} \varphi \\ \sigma \\ \sigma \\ \sigma \end{pmatrix} (W_x x_t + W_h h_{t-1} + b)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot g_t$$
$$h_t = o_t \cdot \phi(c_t)$$

Другая иллюстрация к RNN и LSTM



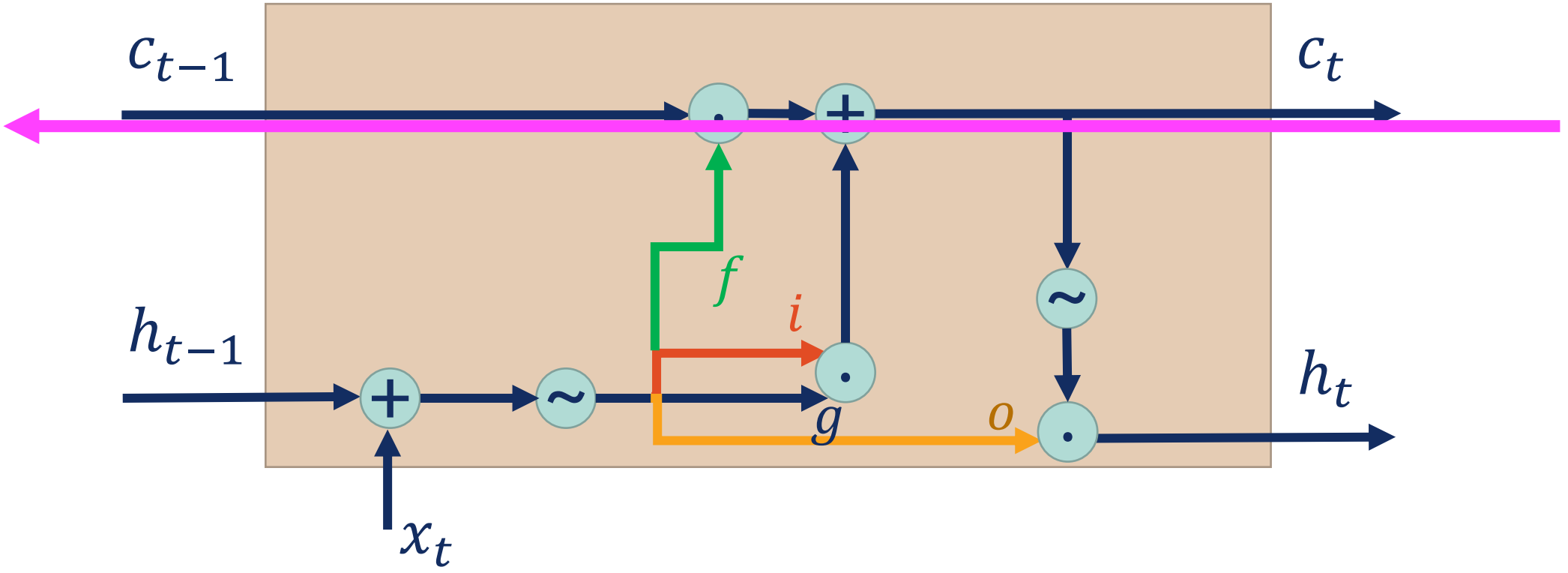
Как решили проблему затухания градиента?



$$\begin{pmatrix} g_t \\ i_t \\ o_t \\ f_t \end{pmatrix} = \begin{pmatrix} \varphi \\ \sigma \\ \sigma \\ \sigma \end{pmatrix} (W_x x_t + W_h h_{t-1} + b)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot g_t$$
$$h_t = o_t \cdot \phi(c_t)$$

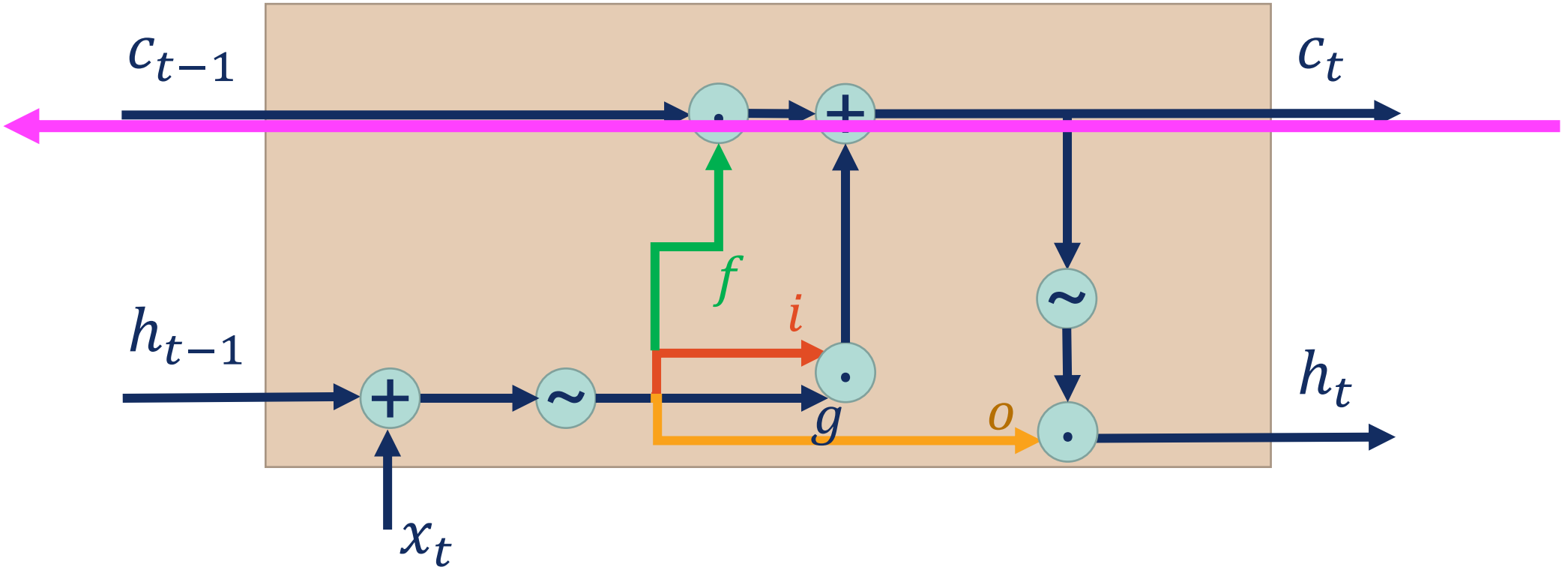
Как решили проблему затухания градиента?



$$\begin{pmatrix} g_t \\ i_t \\ o_t \\ f_t \end{pmatrix} = \begin{pmatrix} \varphi \\ \sigma \\ \sigma \\ \sigma \end{pmatrix} (W_x x_t + W_h h_{t-1} + b)$$

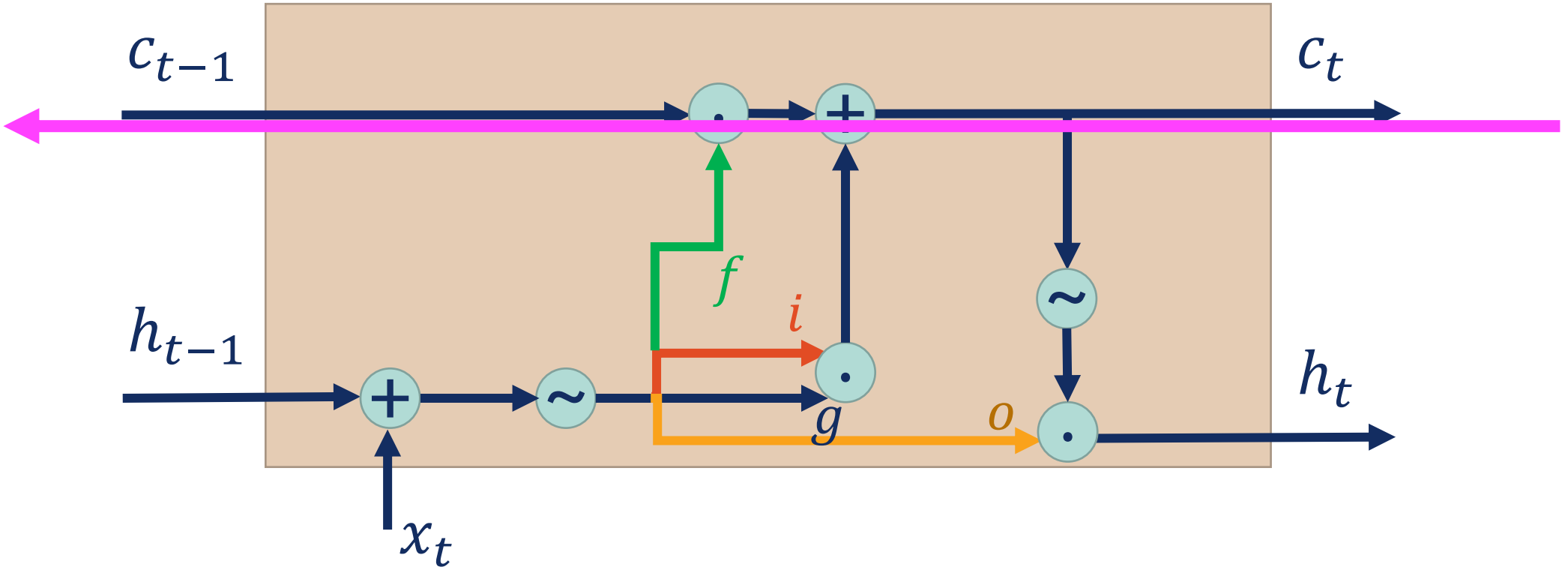
$$c_t = f_t \cdot c_{t-1} + i_t \cdot g_t$$
$$h_t = o_t \cdot \phi(c_t)$$

Как решили проблему затухания градиента?



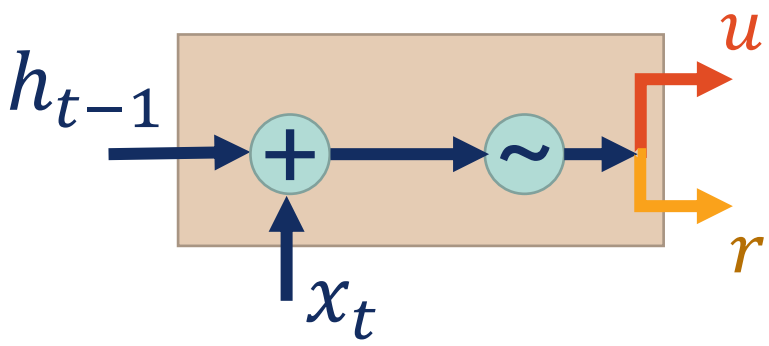
Но нужно, чтобы хотя бы в начале forget gate был открыт.
Вопрос: как этого добиться?

Как решили проблему затухания градиента?

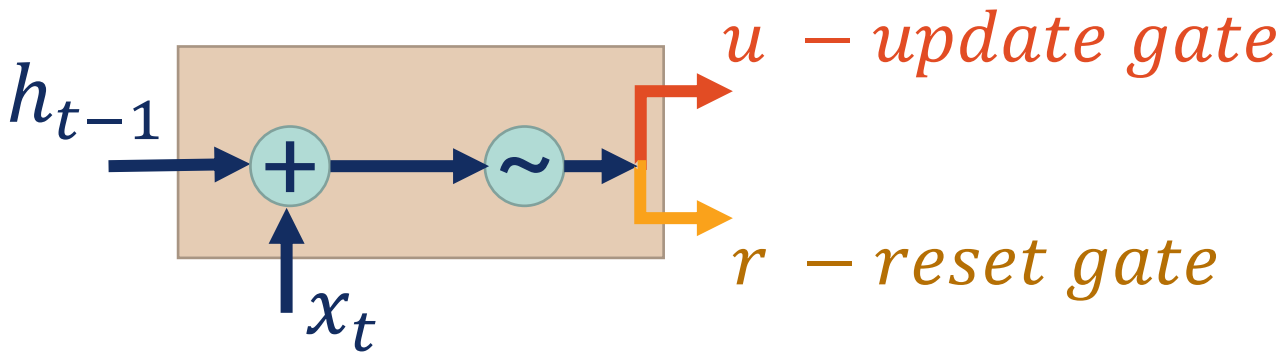


Но нужно, чтобы хотя бы в начале forget gate был открыт.
Вопрос: как этого добиться? (b_f)

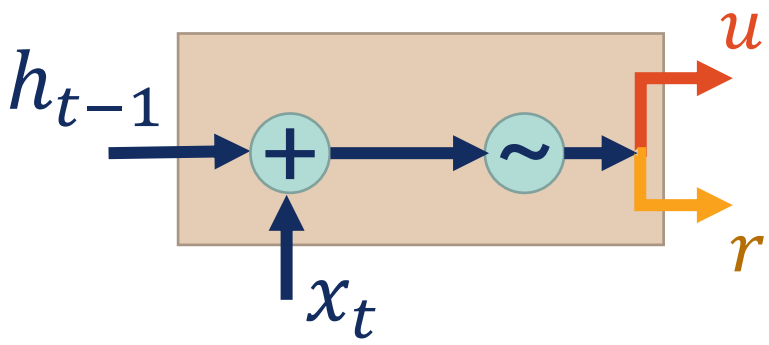
Другой вариант: GRU (Gated Recurrent Unit)



Другой вариант: GRU (Gated Recurrent Unit)

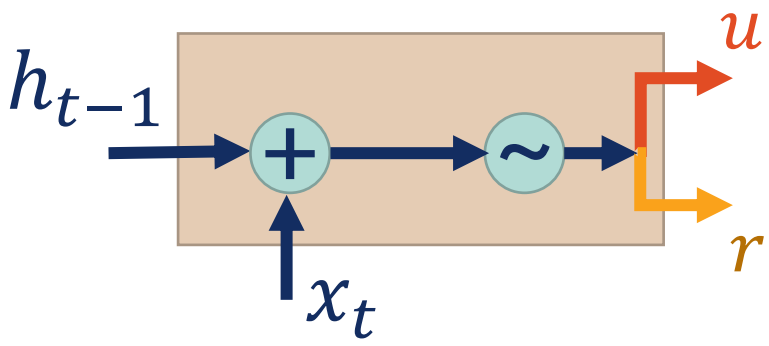


Другой вариант: GRU (Gated Recurrent Unit)



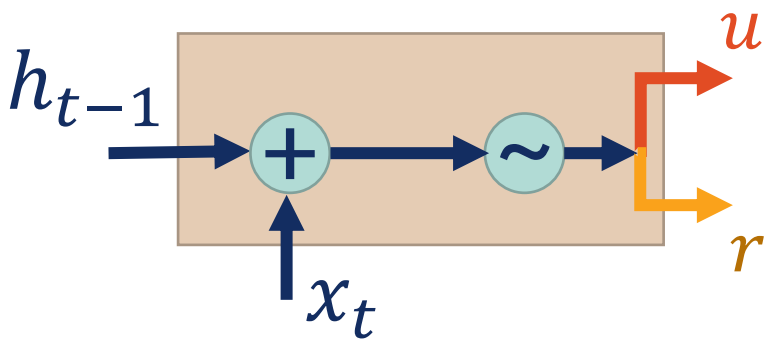
$$\begin{pmatrix} u_t \\ r_t \end{pmatrix} = \sigma(W_x x_t + W_h h_{t-1} + b)$$

Другой вариант: GRU (Gated Recurrent Unit)

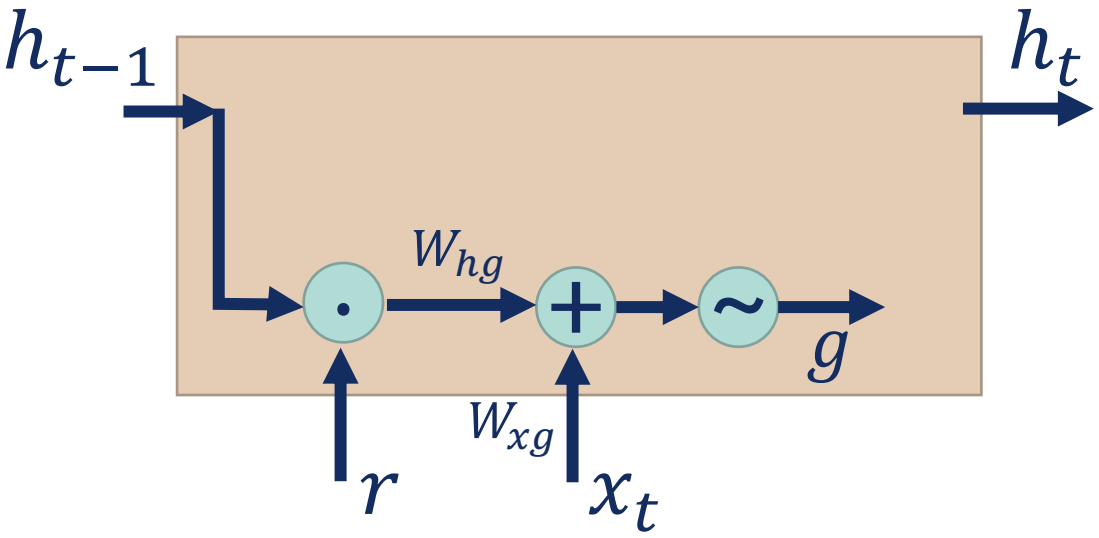


$$\begin{pmatrix} u_t \\ r_t \end{pmatrix} = \sigma(W_x x_t + W_h h_{t-1} + b)$$

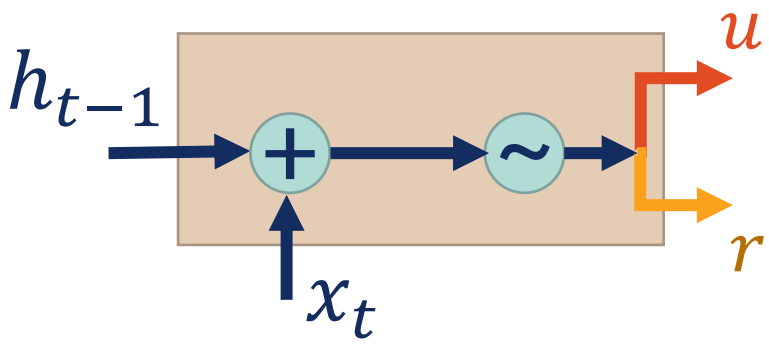
Другой вариант: GRU (Gated Recurrent Unit)



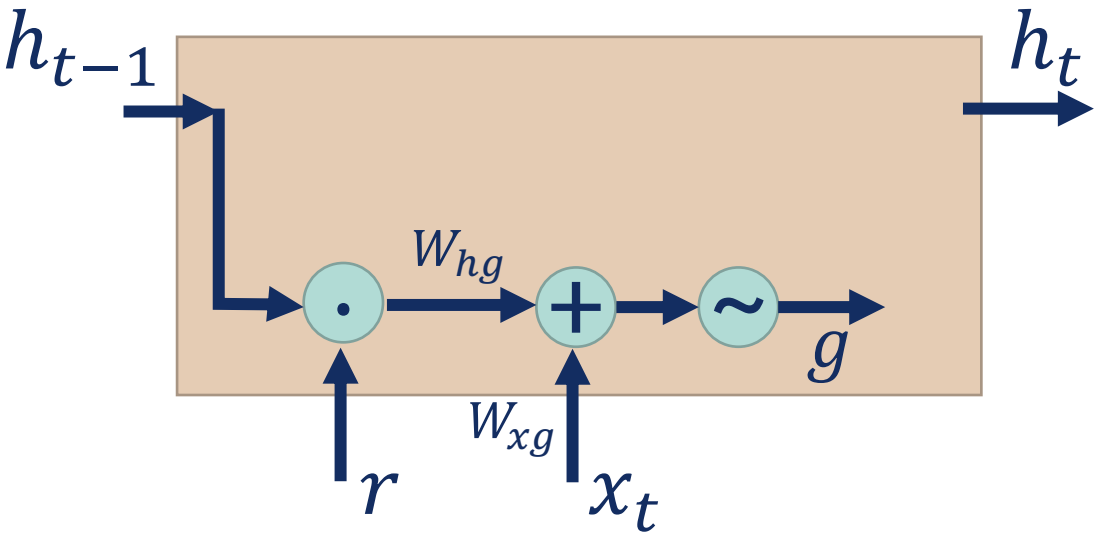
$$\begin{pmatrix} u_t \\ r_t \end{pmatrix} = \sigma(W_x x_t + W_h h_{t-1} + b)$$



Другой вариант: GRU (Gated Recurrent Unit)

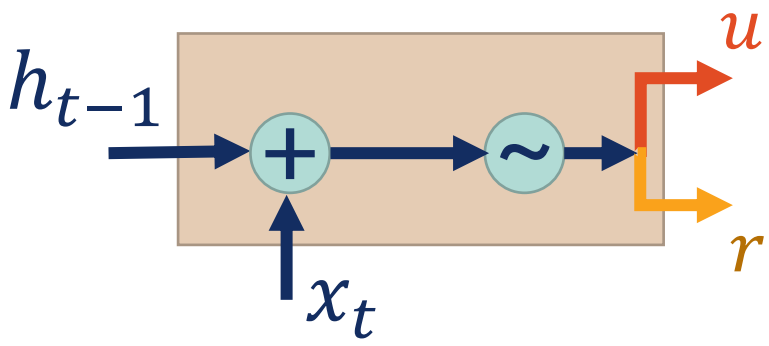


$$\begin{pmatrix} u_t \\ r_t \end{pmatrix} = \sigma(W_x x_t + W_h h_{t-1} + b)$$

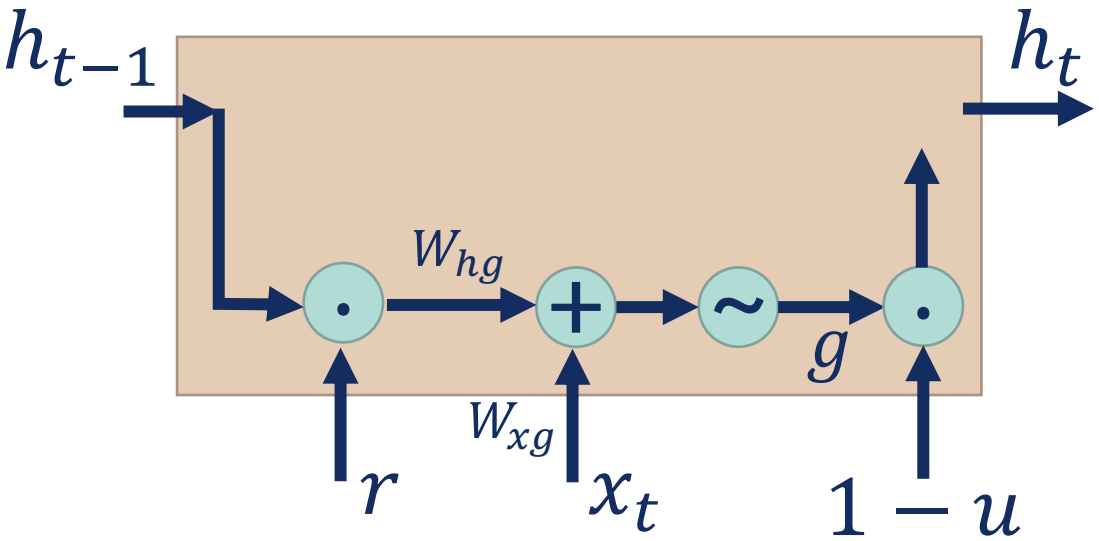


$$g_t = \varphi\left(W_{xg} x_t + W_{hg} r_t \circ h_{t-1} + b_g\right)$$

Другой вариант: GRU (Gated Recurrent Unit)

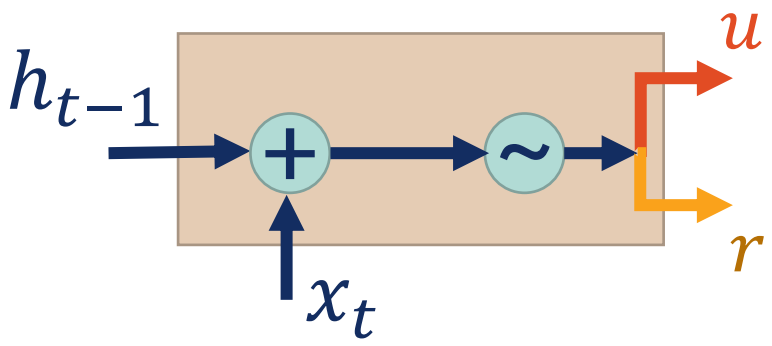


$$\begin{pmatrix} u_t \\ r_t \end{pmatrix} = \sigma(W_x x_t + W_h h_{t-1} + b)$$

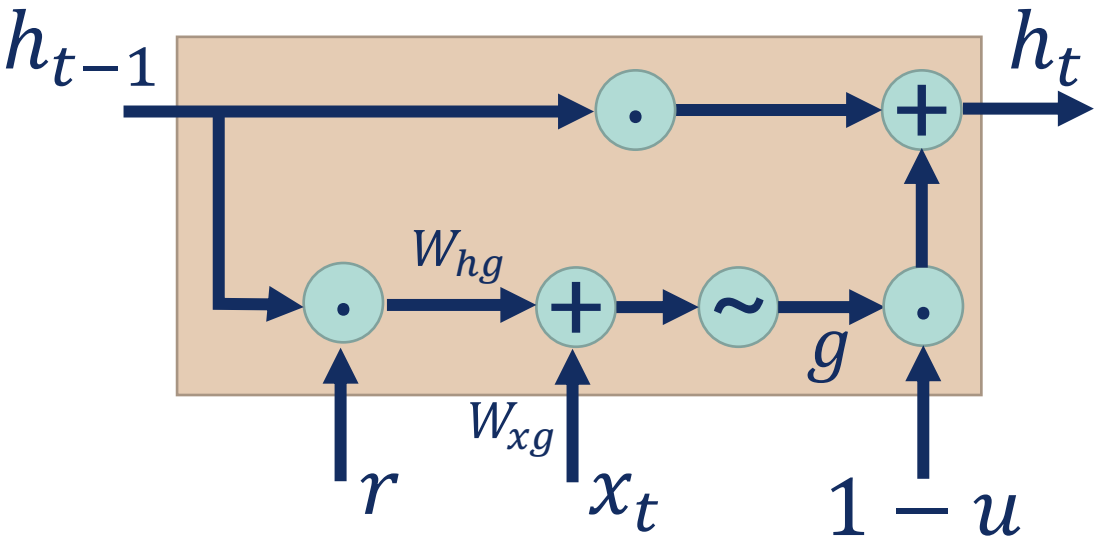


$$g_t = \varphi\left(W_{xg} x_t + W_{hg} r_t \circ h_{t-1} + b_g\right)$$

Другой вариант: GRU (Gated Recurrent Unit)

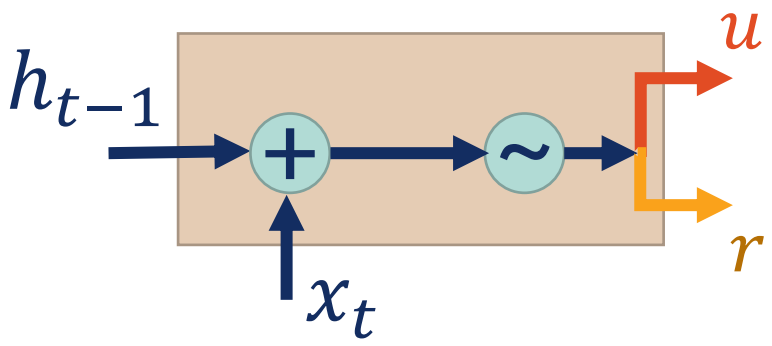


$$\begin{pmatrix} u_t \\ r_t \end{pmatrix} = \sigma(W_x x_t + W_h h_{t-1} + b)$$

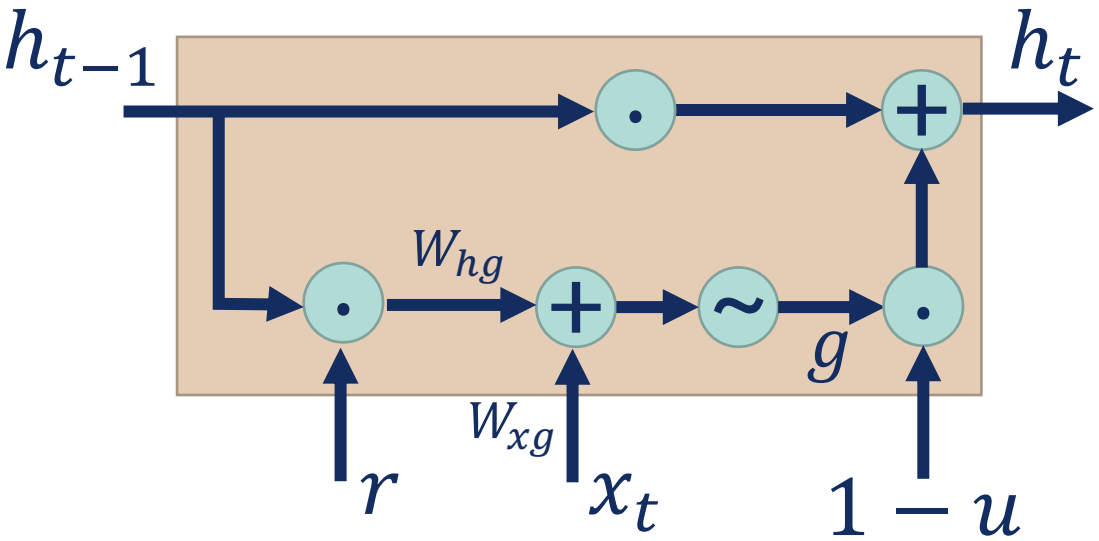


$$g_t = \varphi\left(W_{xg} x_t + W_{hg} r_t \circ h_{t-1} + b_g\right)$$

Другой вариант: GRU (Gated Recurrent Unit)



$$\begin{pmatrix} u_t \\ r_t \end{pmatrix} = \sigma(W_x x_t + W_h h_{t-1} + b)$$



$$g_t = \varphi\left(W_{xg} x_t + W_{hg} r_t \circ h_{t-1} + b_g\right)$$

$$h_t = (1 - u_t) \cdot g_t + u_t \cdot h_{t-1}$$

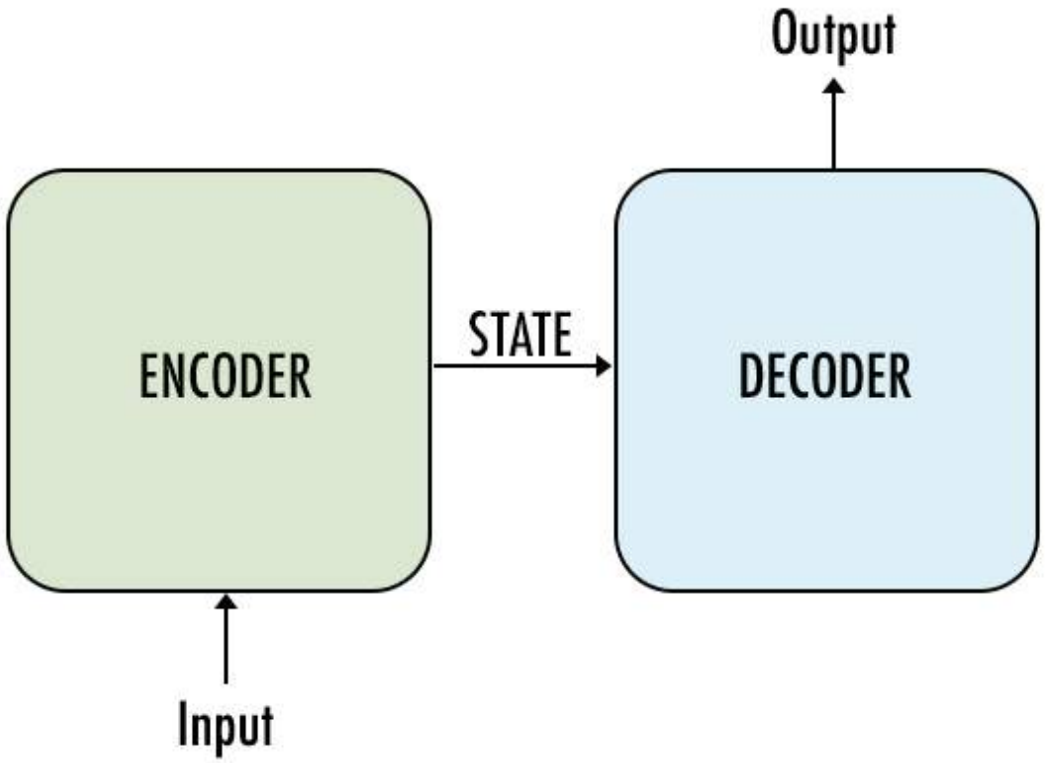
LSTM или GRU

В GRU меньше параметров, чем в LSTM, поэтому:

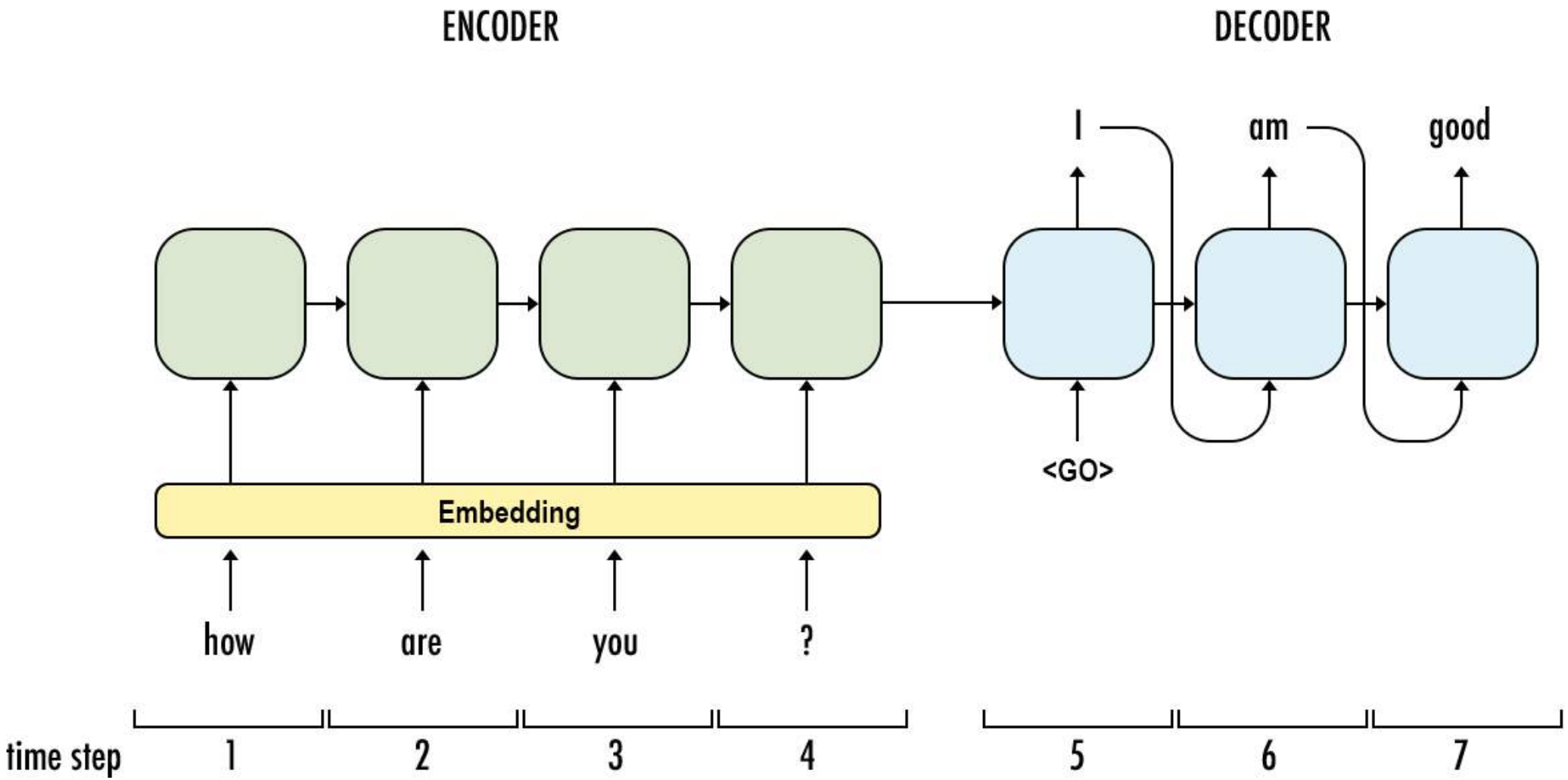
- GRU менее склонна к переобучению
- LSTM (если не переобучится) может настраиваться более точно

4. Seq2seq & attention

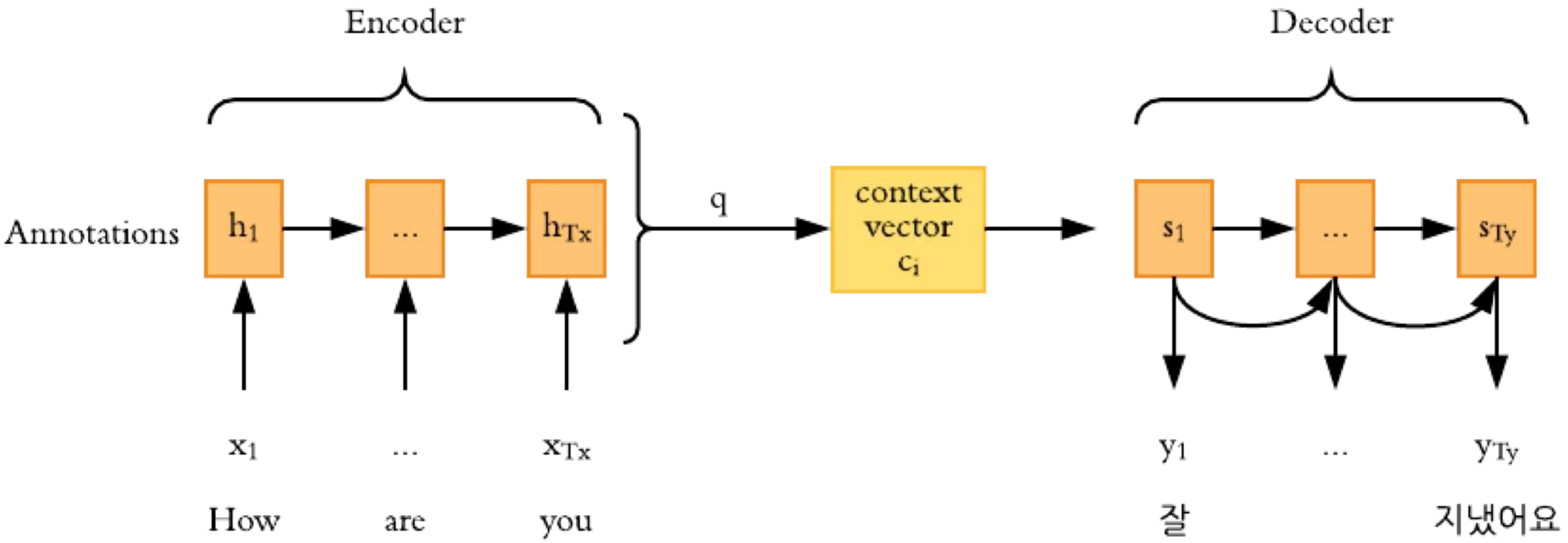
Seq2seq и архитектура encoder-decoder



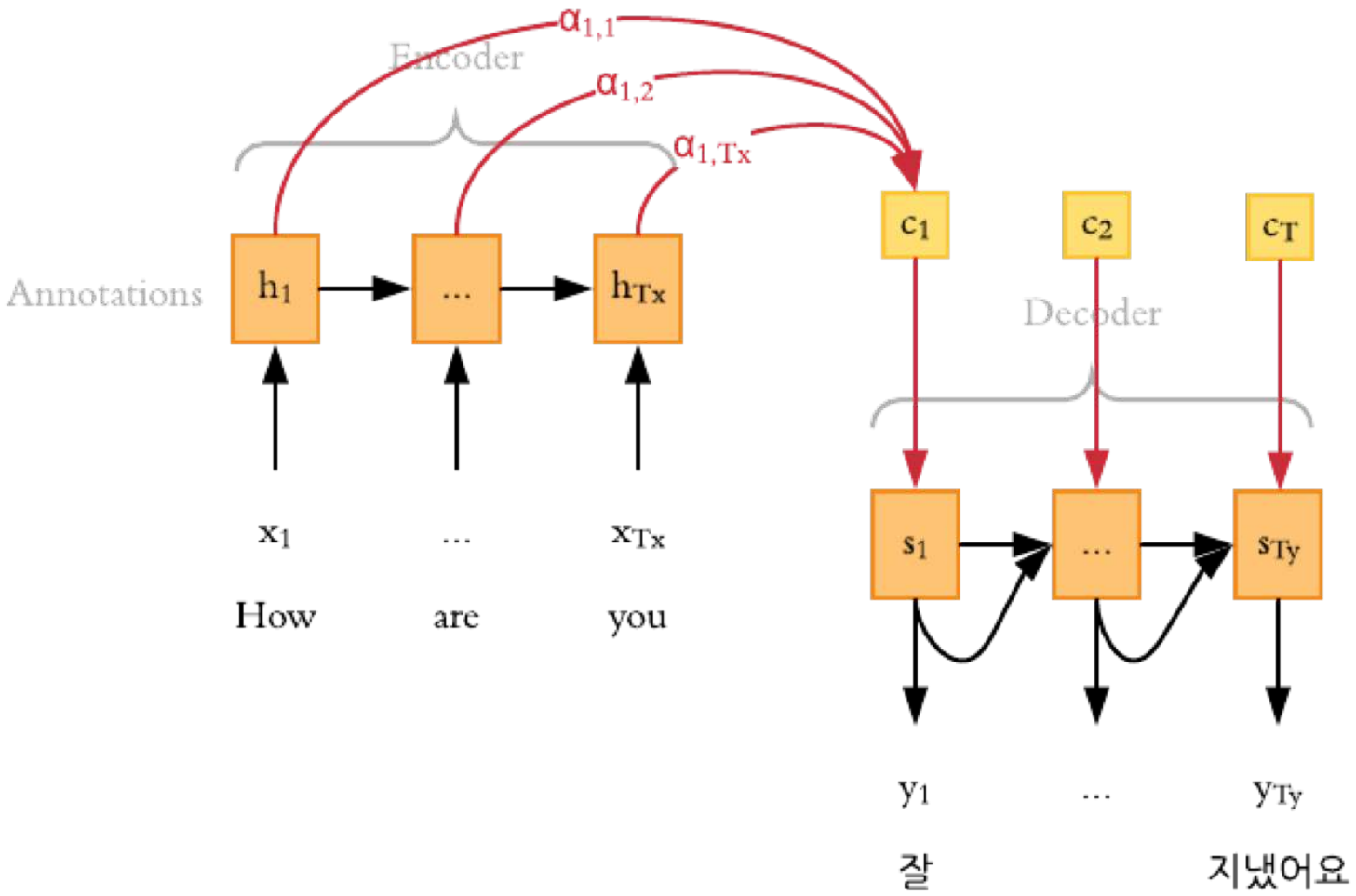
Seq2seq и архитектура encoder-decoder



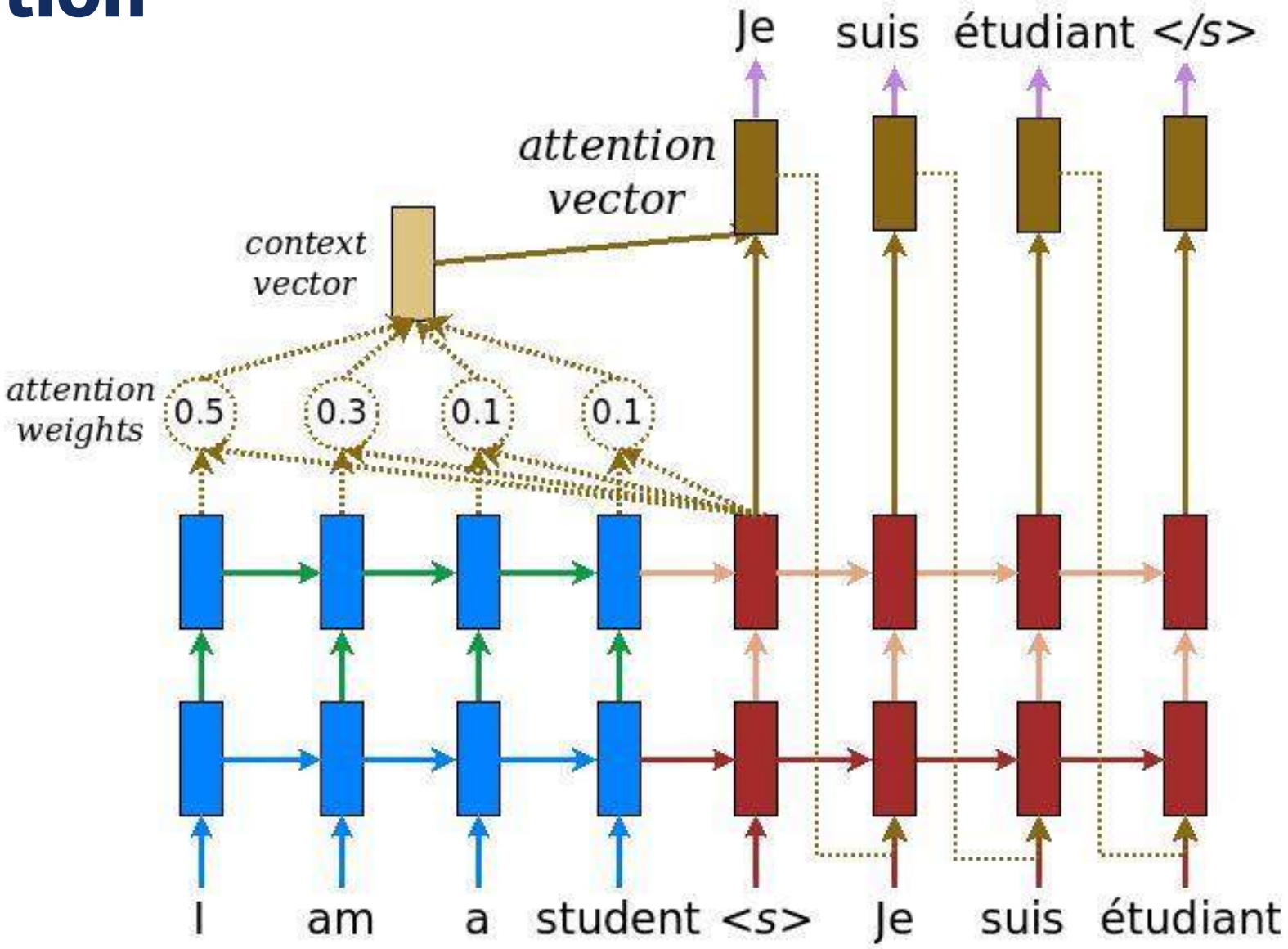
Context/thought vector



Attention


Encoder
Decoder
Annotations
$\alpha_{1,1}$
$\alpha_{1,2}$
$\alpha_{1,Tx}$
h_1
...
h_{Tx}
c_1
c_2
c_T
s_1
...
s_{Ty}
x_1
...
x_{Tx}
How
are
you
y_1
...
y_{Ty}
잘
지냈어요

Attention


Je
suis
étudiant </s>
attention vector
context vector
attention weights
0.5
0.3
0.1
0.1
I
am
a
student <s>
Je
suis
étudiant

Self-attention & transformer

"Attention Is All You Need" [Vaswani, 2017]

План

1. Напоминание
2. Vanilla RNN
3. LSTM и GRU
4. Seq2seq & attention

Data Mining in Action

Лекция 10

Группа курса в Telegram:

https://t.me/joinchat/B1OlTk74nRV56Dp1TDJGNA