

Лекция 6

Временные ряды и рекуррентные нейронные сети

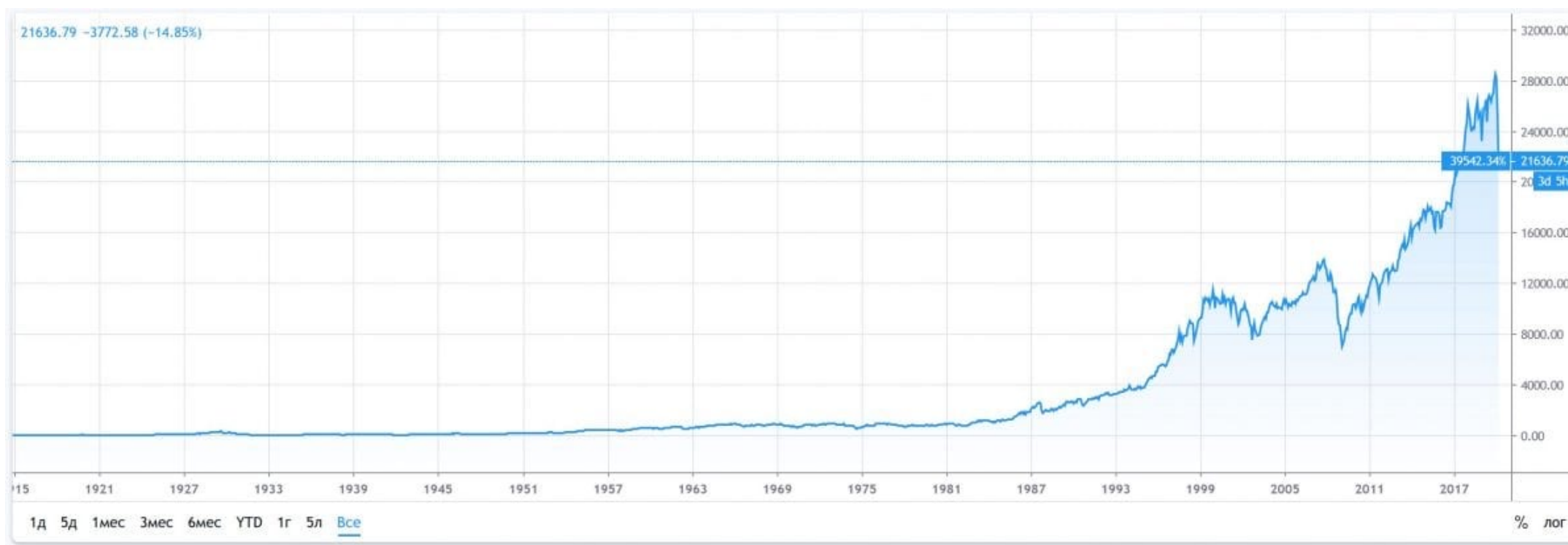
Разработка нейросетевых систем

Канев Антон Игоревич

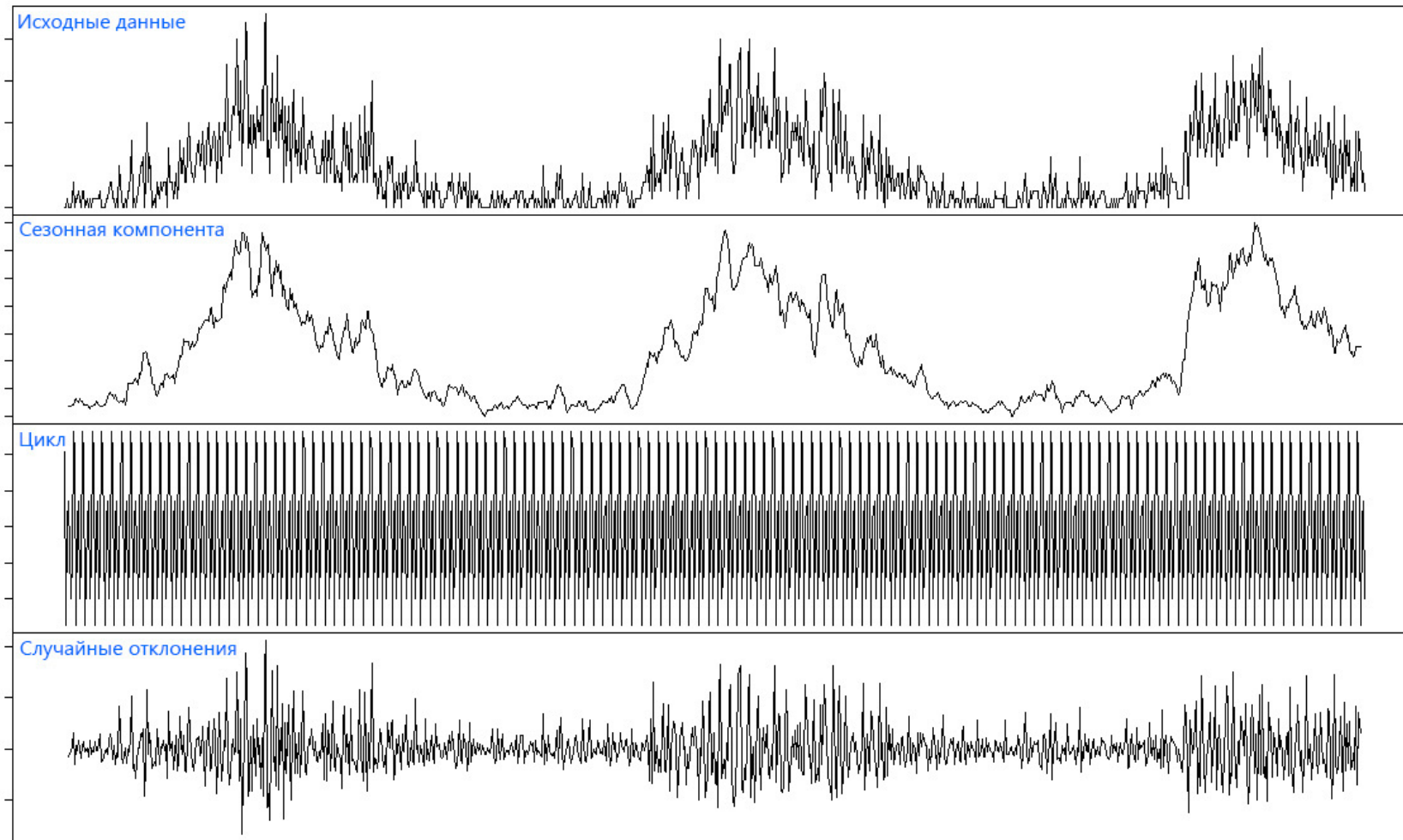
Временной ряд

- **Временной ряд** — собранный в разные моменты времени статистический материал о значении каких-либо параметров.
- Временной ряд существенно отличается от простой выборки данных, так как при анализе учитывается взаимосвязь измерений со временем, а не только статистическое разнообразие и статистические характеристики выборки.
- Стационарный временной ряд – ряд, у которого не меняются математическое ожидание и дисперсия со временем.

Стационарный временной ряд



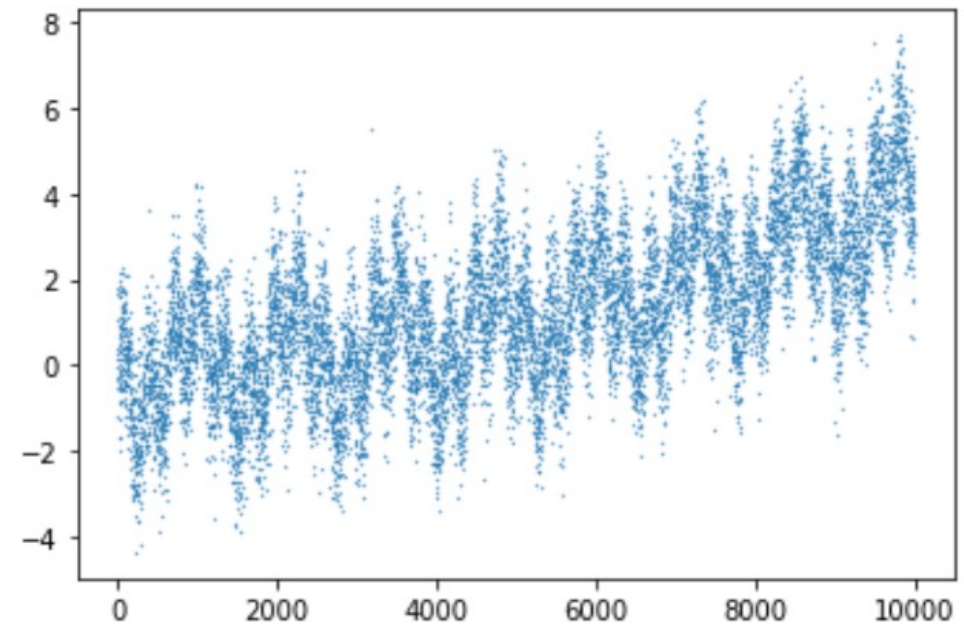
Преобразование временных рядов



Синтетические данные

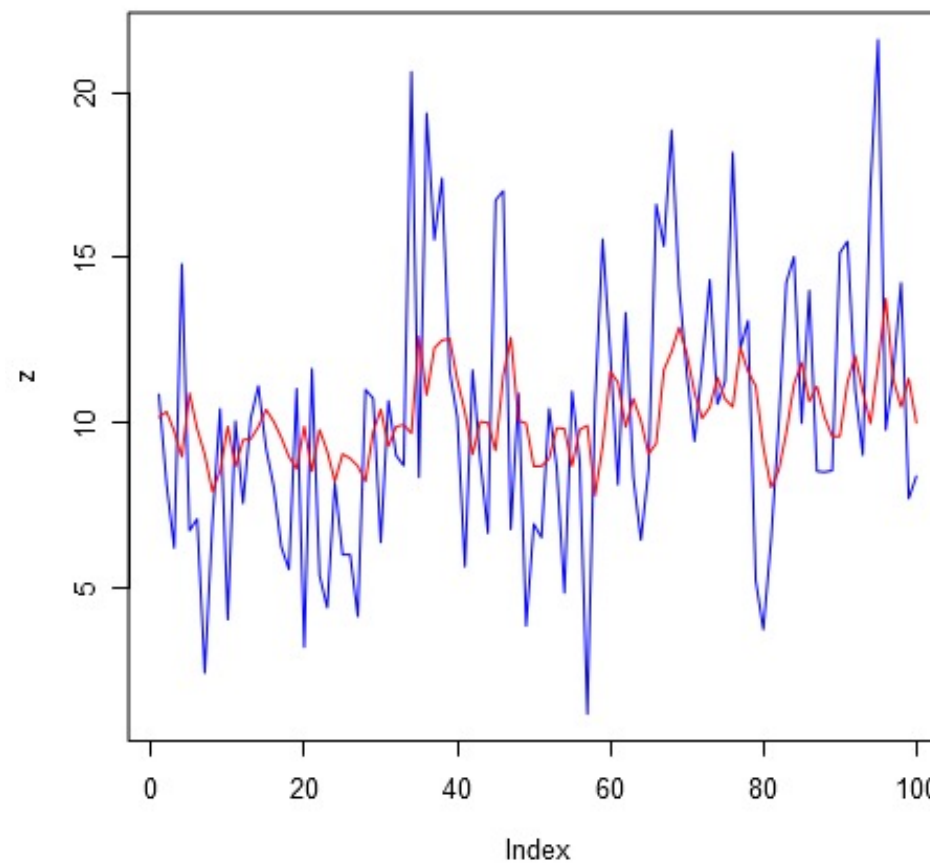
```
X = np.arange(10000)
y = np.sin(X/50)-np.sin(X/200)+(2*X/X.size)**2
y += np.random.normal(scale=1.0, size=y.size)
plt.scatter(X, y[:10000], s=0.1)

df = pd.Series(y)
df = df.diff().dropna()
```



Модель авторегрессии

$$X_t = c + \sum_{i=1}^p a_i X_{t-i} + \varepsilon_t,$$

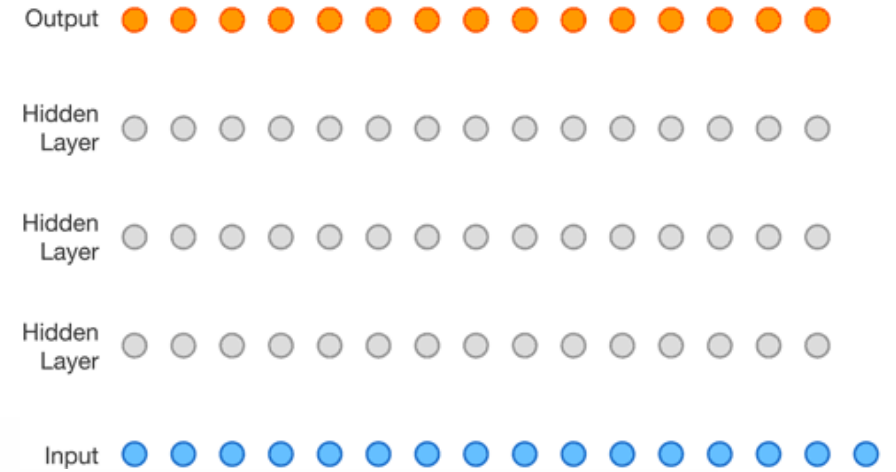
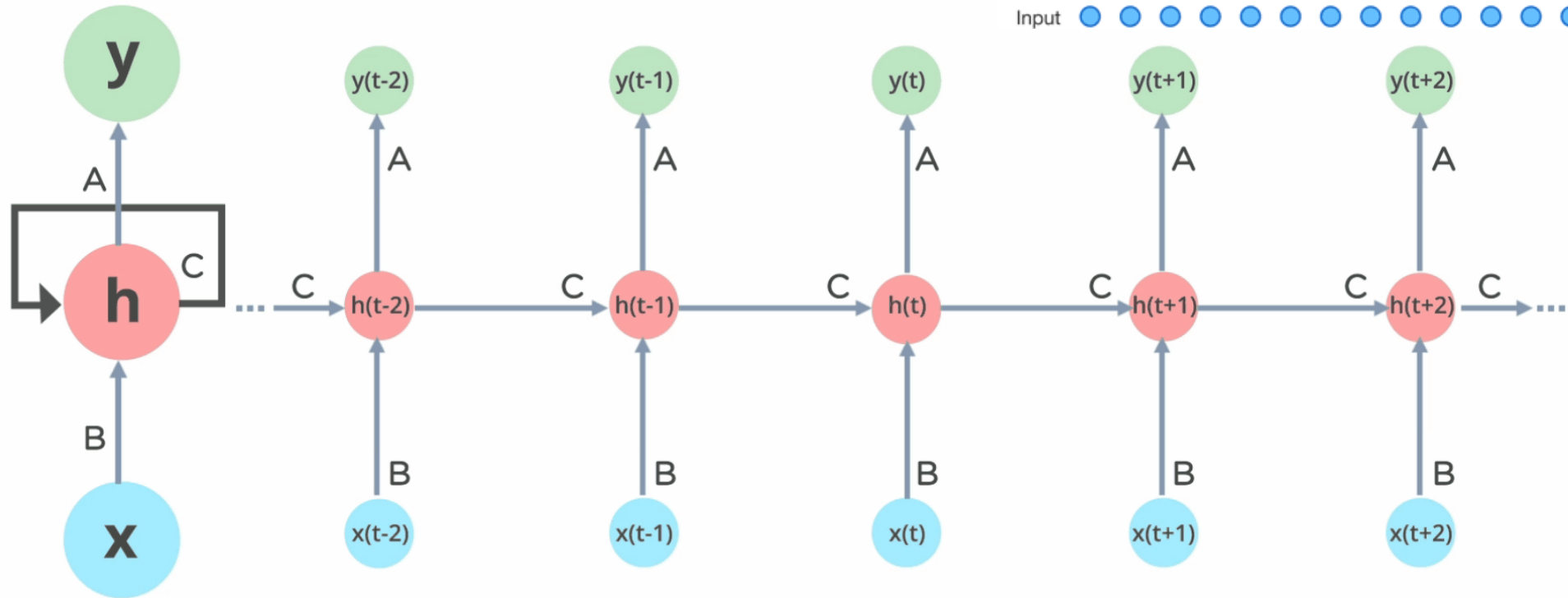


Рекуррентная нейронная сеть

- **Рекуррентные нейронные сети** (*Recurrent neural network; RNN*) — вид нейронных сетей, где связи между элементами образуют направленную последовательность.
- Благодаря этому появляется возможность обрабатывать серии событий во времени или последовательные пространственные цепочки.
- В отличие от многослойных перцептронов, рекуррентные сети могут использовать свою внутреннюю память для обработки последовательностей произвольной длины.

Рекуррентная нейросеть

- Рекуррентная нейросеть позволяет реализовать память
- GRU, LSTM



Рекуррентная нейронная сеть

- Сеть Элмана

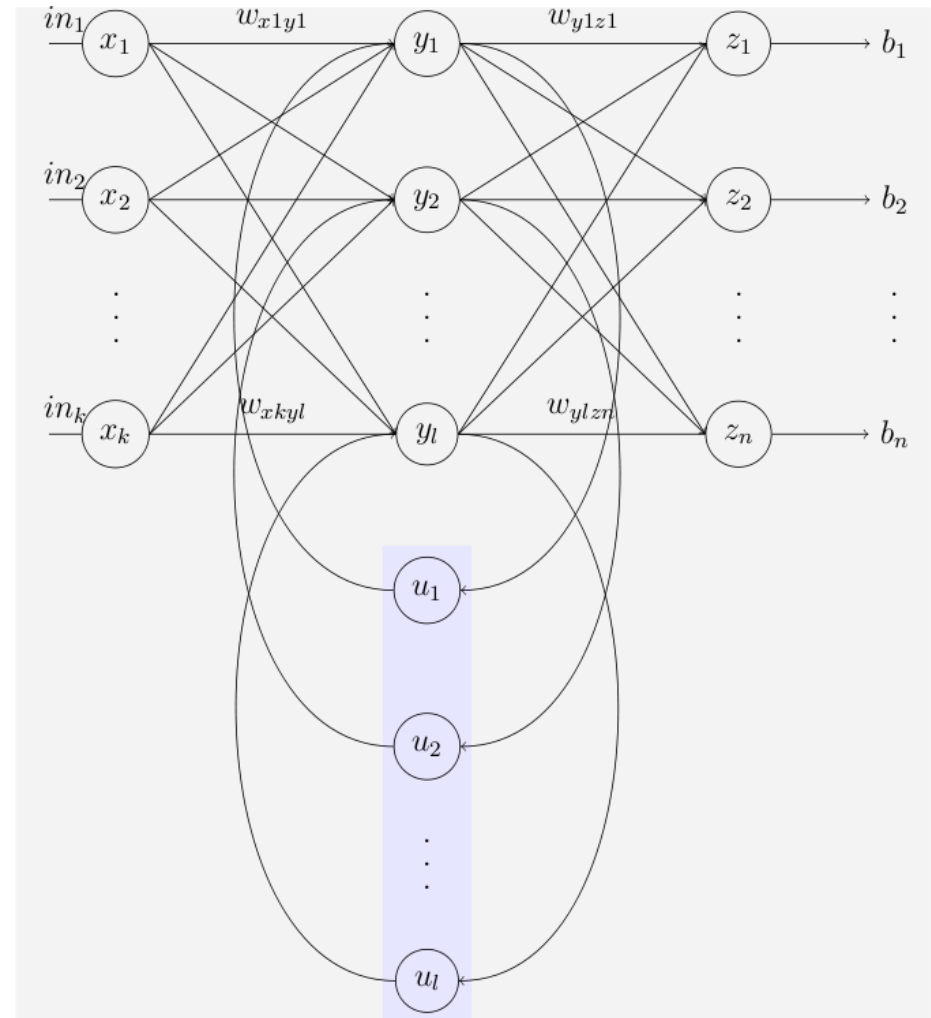
$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$

$$y_t = \sigma_y(W_y h_t + b_y)$$

- Сеть Джордана

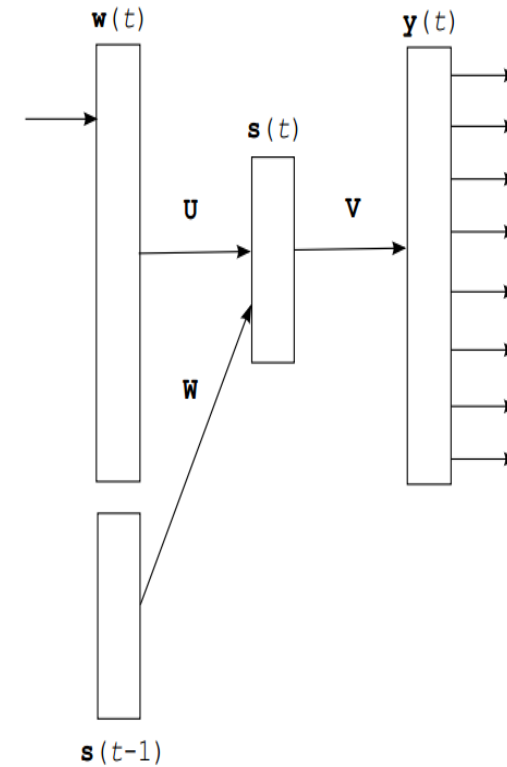
$$h_t = \sigma_h(W_h x_t + U_h y_{t-1} + b_h)$$

$$y_t = \sigma_y(W_y h_t + b_y)$$



Архитектура: простая RNN

- Входной слой, скрытый слой с рекуррентными соединениями и выходной слой
- В теории скрытый слой может обладать неограниченной памятью
- Также называется сетью Элмана (*Finding structure in time*, Elman 1990)



Архитектура: простая RNN

$$\mathbf{s}(t) = f(\mathbf{U}\mathbf{w}(t) + \mathbf{W}\mathbf{s}(t-1))$$

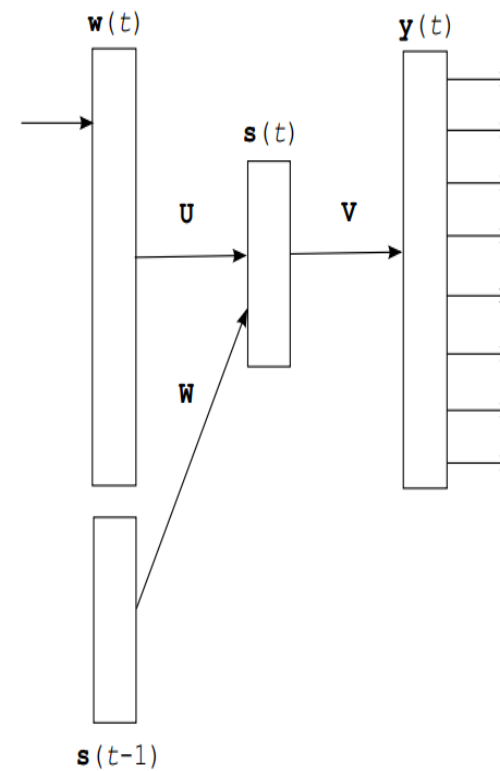
$$\mathbf{y}(t) = g(\mathbf{V}\mathbf{s}(t))$$

$f()$ чаще всего сигмоидальная
активационная функция:

$$f(z) = \frac{1}{1 + e^{-z}}$$

$g()$ чаще всего softmax:

$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$$



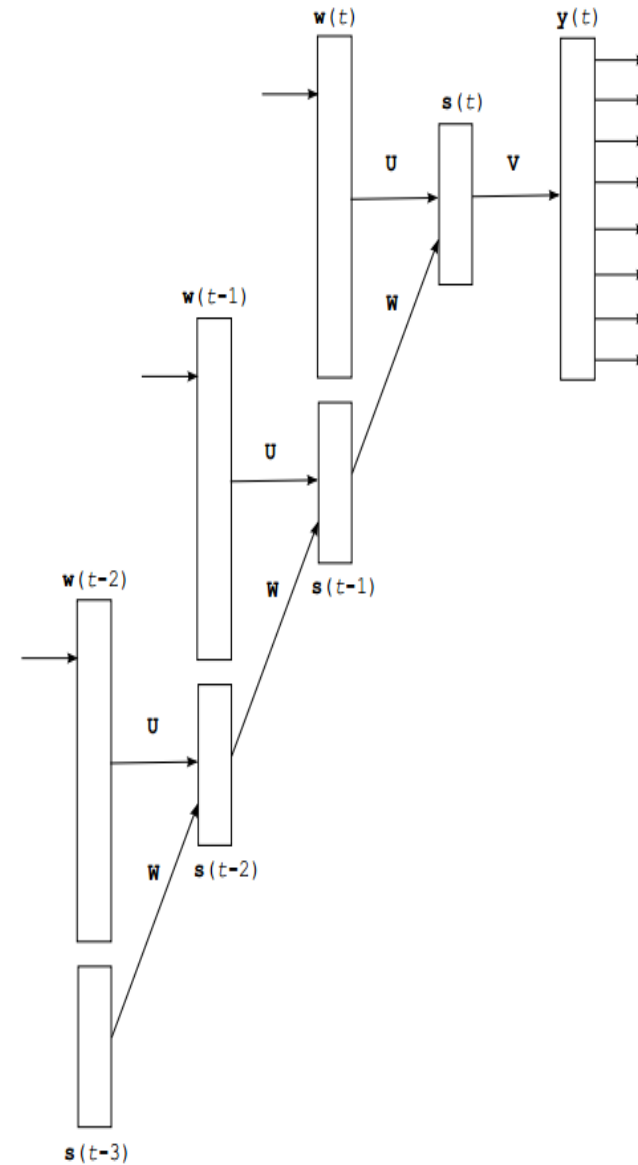
Обучение: обратное распространение по времени

(Backpropagation through time, BPTT)

- Как обучать рекуррентные сети?
- Выходное значение зависит от состояния скрытого слоя, который зависит от всех предыдущих состояний скрытого слоя (и, следовательно, всех предыдущих входов)
- Рекуррентная сеть может рассматриваться как (очень глубокая) сеть прямого распространения с общими весами

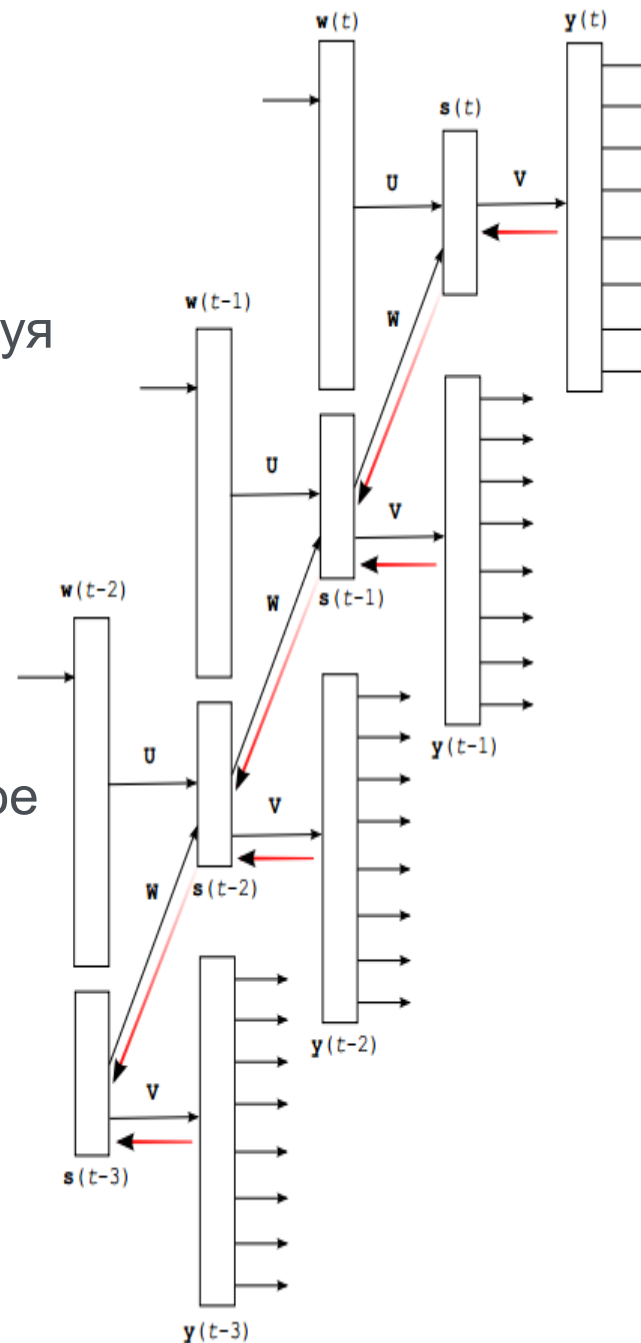
Обратное распространение по времени

- Идея заключается в том, что РНС разворачивается во времени
- Мы получаем глубокую нейронную сеть с общими весами **U** и **W**
- Часто бывает достаточно развернуть на несколько шагов (называется *укороченным BPTT*)



Обратное распространение по времени

- Мы тренируем развернутый РНС, используя обычное обр. распр-е + СГС
- На практике ограничивают количество шагов разворачивания до 5 – 10
- Эффективнее вычислить градиент после нескольких обучающих примеров (пакетное обучение)



Исчезающие градиенты

- Когда мы распространяем градиенты назад во времени, обычно их величина быстро уменьшается: это называется «проблемой исчезающего градиента»
- На практике это означает, что изучение долгосрочных зависимостей в данных затруднительно для простой RNN архитектуры
- Специальные архитектуры RNN решают эту проблему:
 - Экспоненциальная память трассировки (Jordan 1987, Mozer 1989)
 - Долгая краткосрочная память (Hochreiter & Schmidhuber, 1997))
 - будут обсуждены во второй части этой лекции

Обучение: Исчезающие градиенты

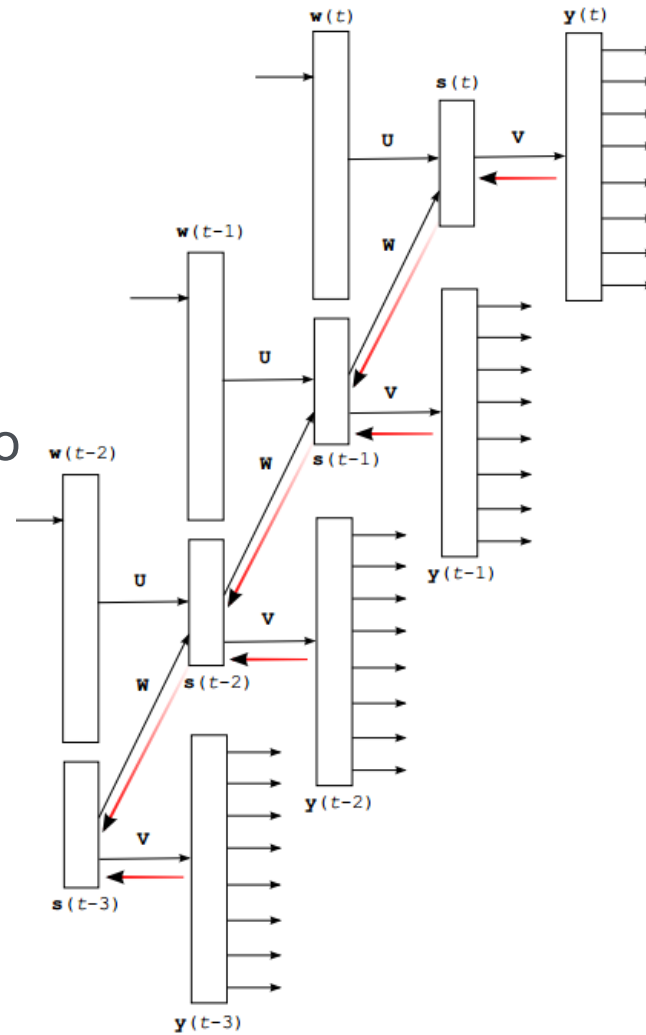
- Множество научных теорий
 - лучше инициализировать рекуррентную матрицу и использовать импульс во время обучения
- Sutskever et.al.,: On The Importance of Initialization and Momentum in Deep Learning
 - изменять архитектуру

Взрывающиеся градиенты

- Иногда градиенты начинают экспоненциально возрастать во время обратного распространения через рекуррентные веса
- Бывает редко, но эффект может быть катастрофическим: огромные градиенты приведут к большому изменению весов и, таким образом, разрушат то, что было изучено до сих пор
- Одна из основных причин, по которым RNN должны были быть нестабильными
- Простое решение (впервые опубликован в RNNLM toolkit в 2010): обрезать или нормализовать значения градиентов, чтобы избежать огромных изменений весов

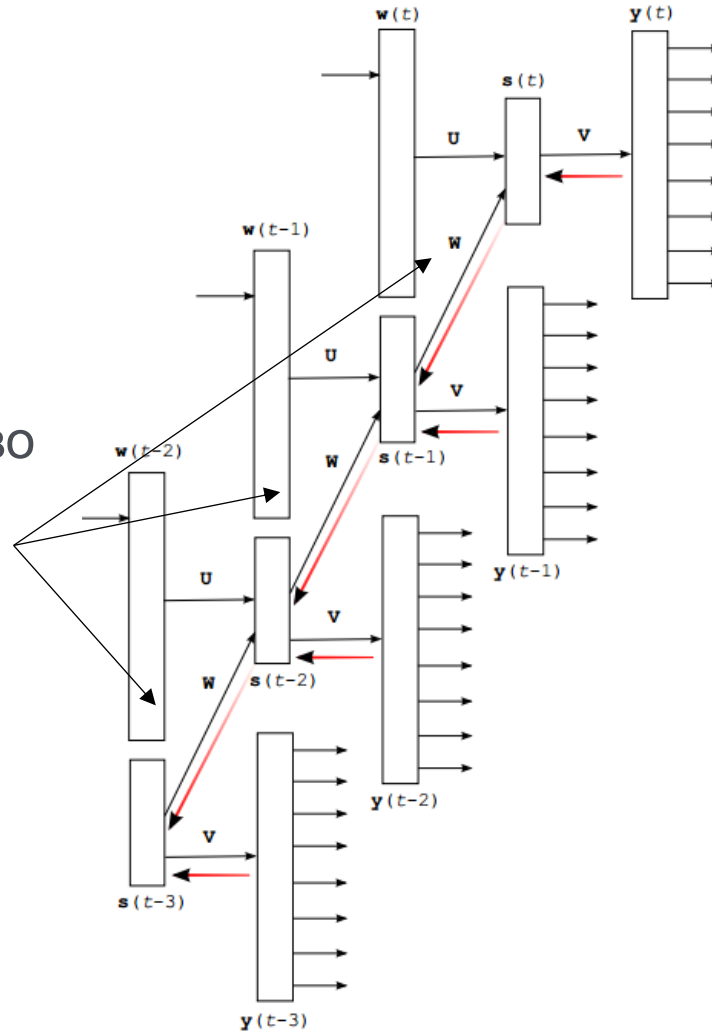
Обучение: Взрывающиеся градиенты

- Обрезка градиента во время обратного распространения по времени



Обучение: Взрывающиеся градиенты

- Обрезка градиента во время обратного распространения по времени



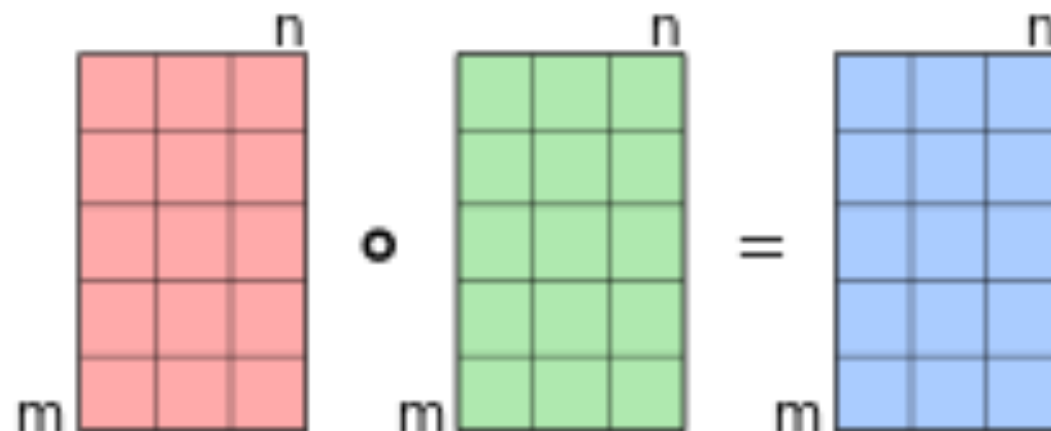
Долгая краткосрочная память (LSTM)

- Недавно приобрела большую популярность
- Имейте явные «ячейки» памяти для хранения кратковременных активаций, наличие дополнительных вентилях частично устраняет проблему исчезающего градиента
- Многослойные версии показали, что они хорошо работают над задачами, которые имеют среднесрочные зависимости

Произведение Адамара

- Произведение Адамара оперирует двумя матрицами одинаковой размерности и создаёт новую матрицу идентичной размерности.

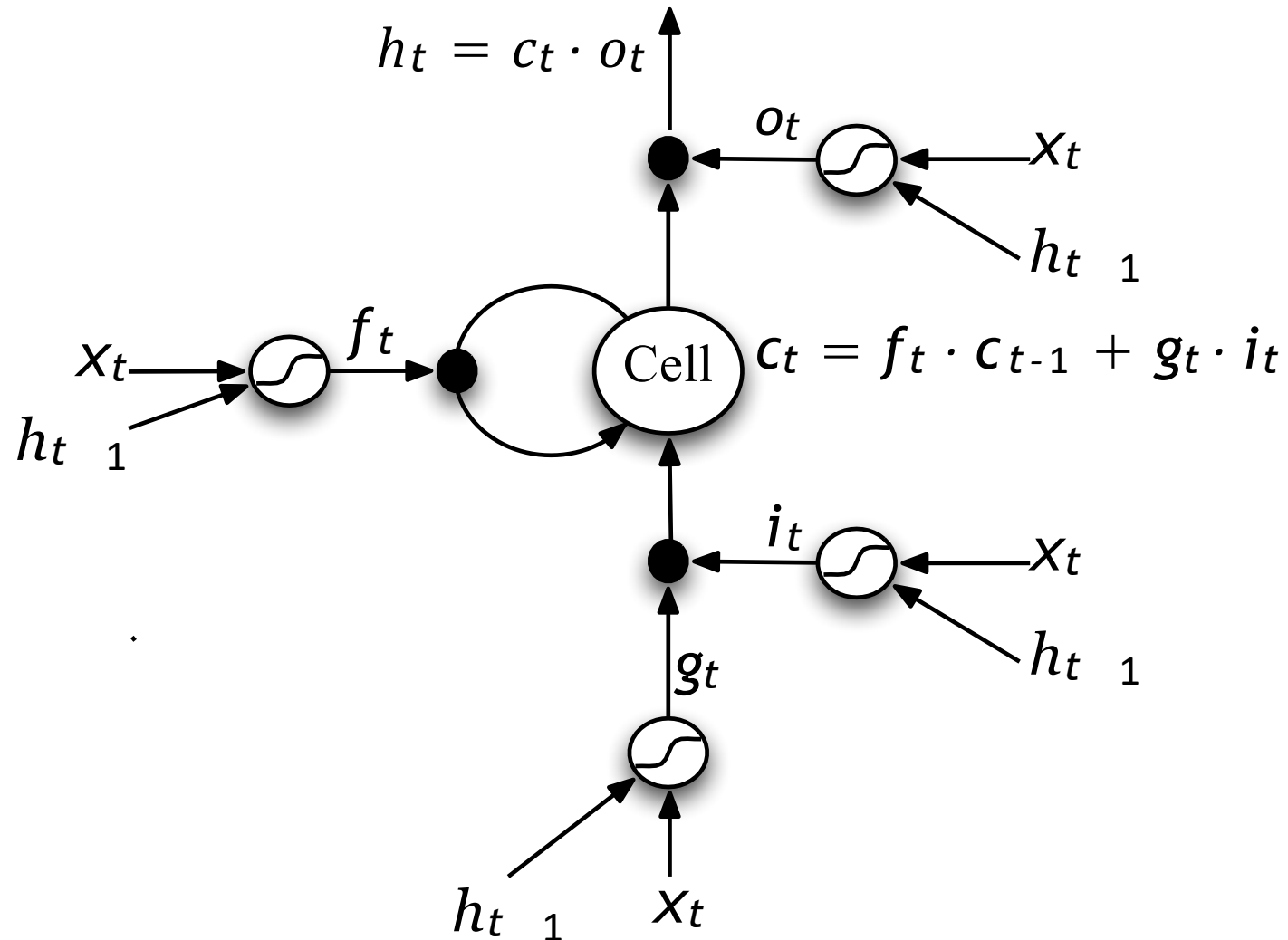
$$(A \circ B)_{i,j} = (A)_{i,j} \cdot (B)_{i,j}$$



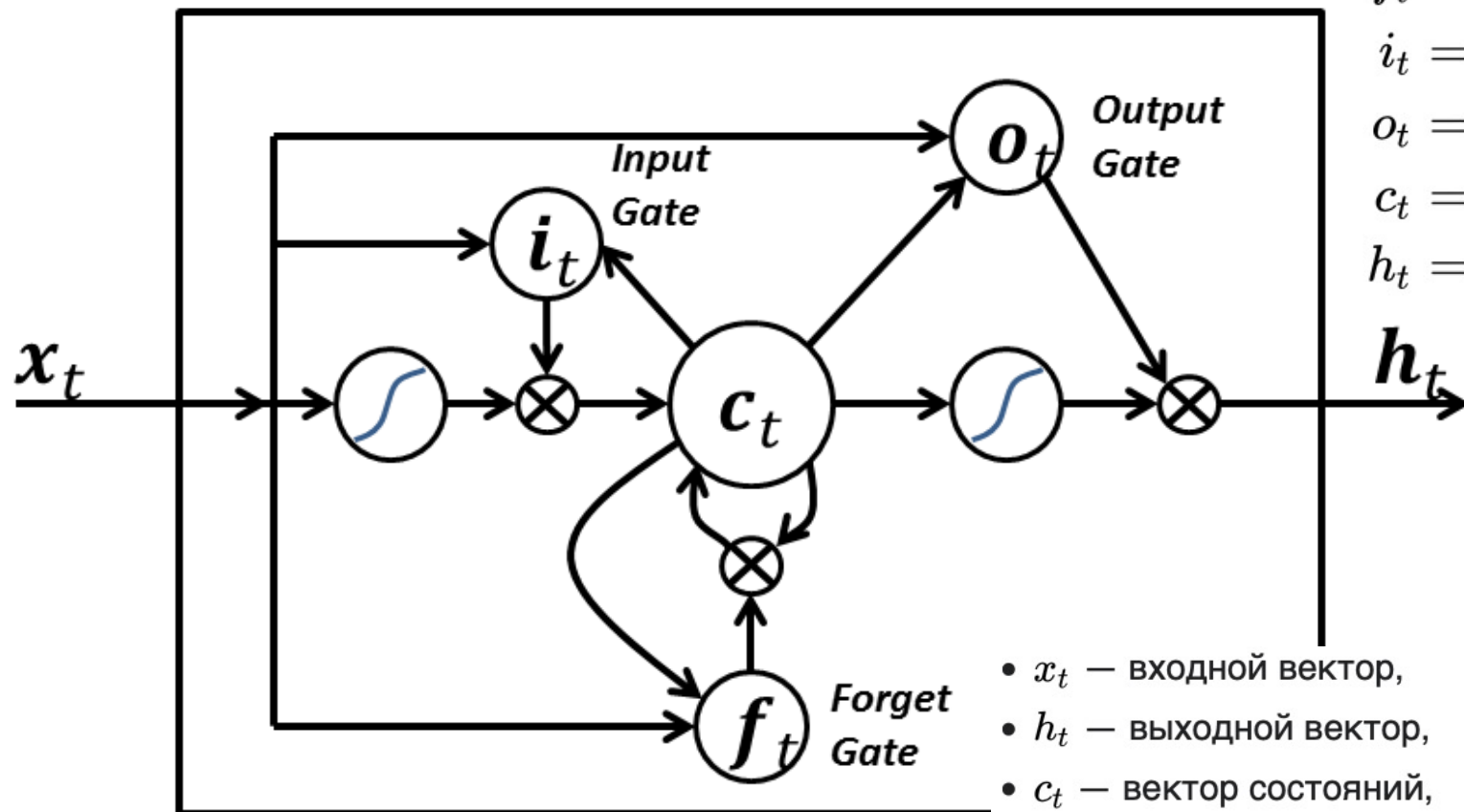
$$A \circ B = C$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \circ \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11} b_{11} & a_{12} b_{12} & a_{13} b_{13} \\ a_{21} b_{21} & a_{22} b_{22} & a_{23} b_{23} \\ a_{31} b_{31} & a_{32} b_{32} & a_{33} b_{33} \end{bmatrix}$$

Долгая краткосрочная память (LSTM)



LSTM



$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$

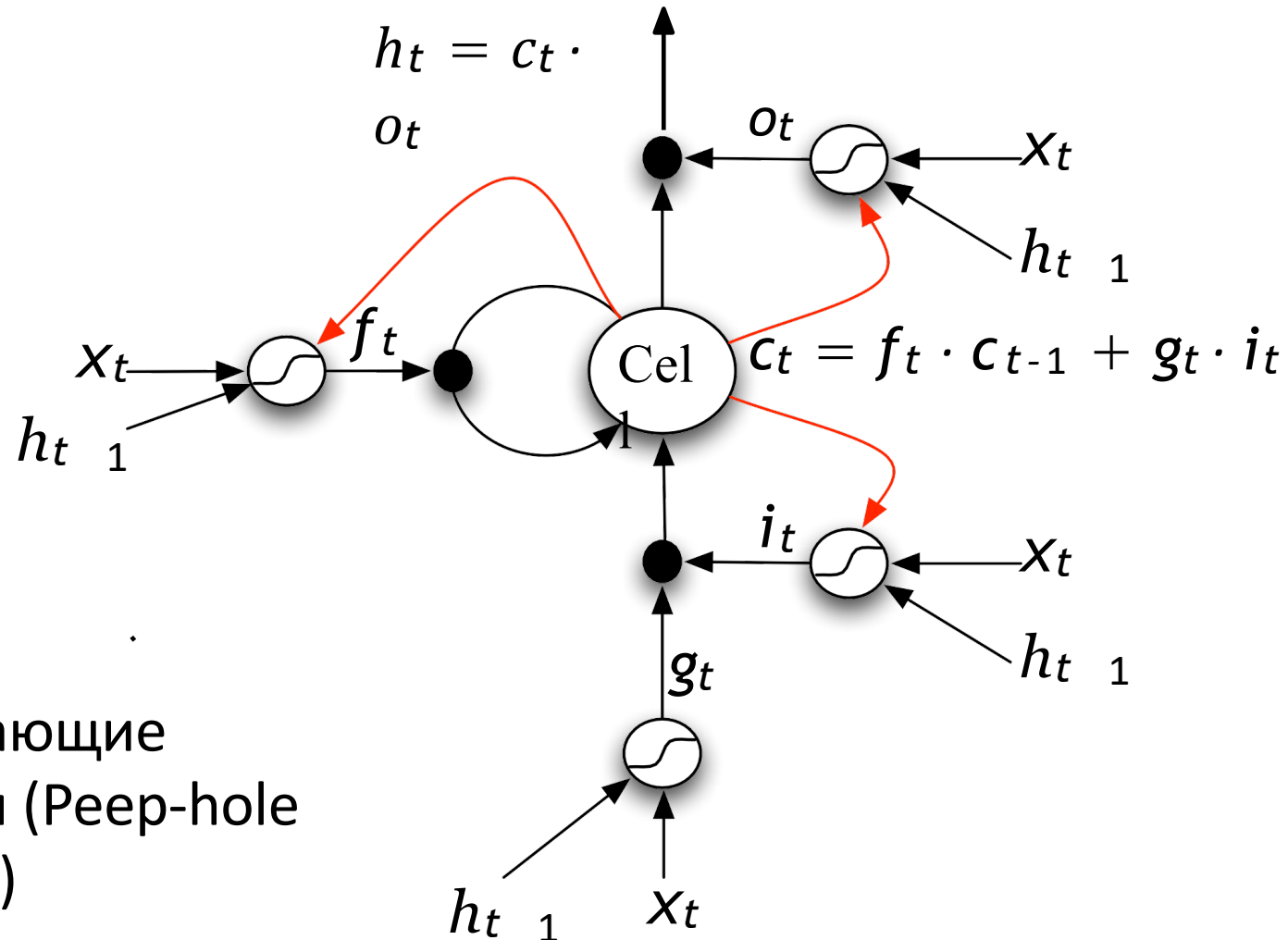
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

$$h_t = o_t \circ \sigma_h(c_t)$$

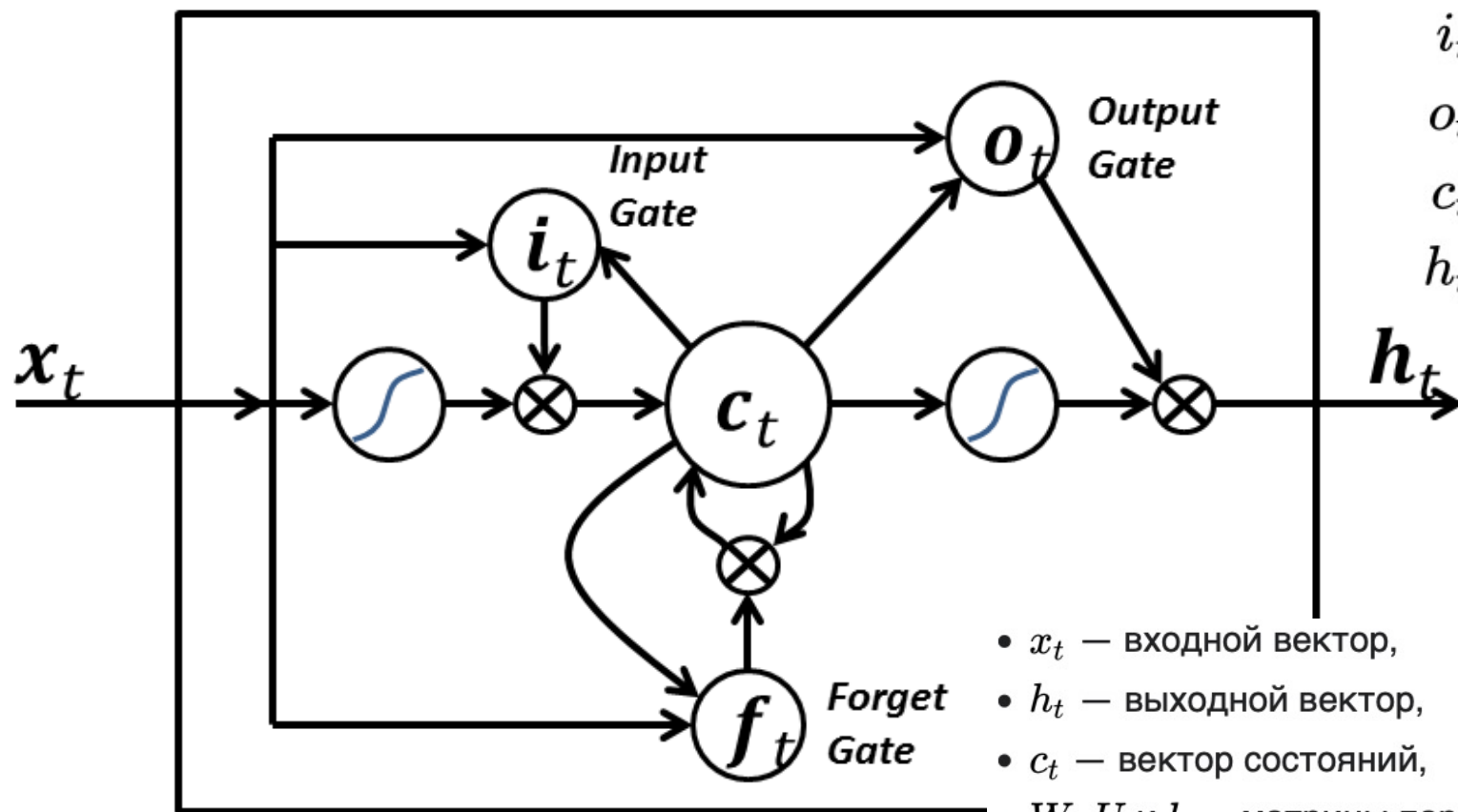
- x_t — входной вектор,
- h_t — выходной вектор,
- c_t — вектор состояний,
- W, U и b — матрицы параметров и вектор,
- f_t, i_t и o_t — векторы вентиляей,
 - f_t — вектор вентиля забывания, вес запоминания старой информации,
 - i_t — вектор входного вентиля, вес получения новой информации,
 - o_t — вектор выходного вентиля, кандидат на выход.

Долгая краткосрочная память (LSTM) с «глазками»



Подглядывающие
соединения (Peep-hole
connections)

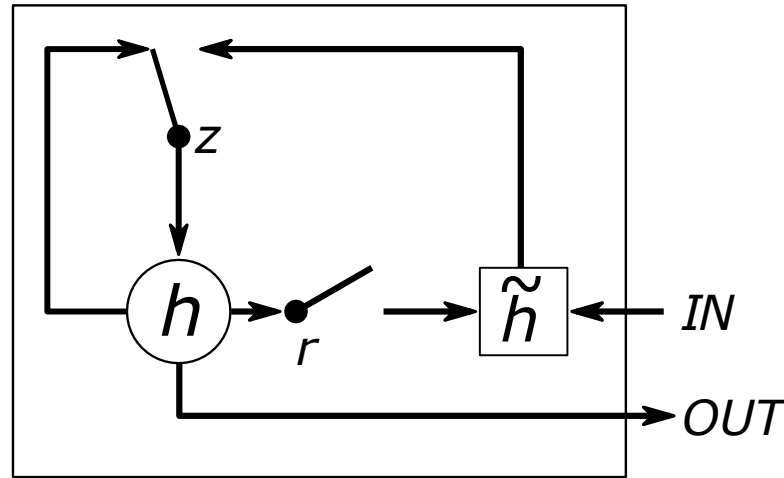
LSTM



$$\begin{aligned}f_t &= \sigma_g(W_f x_t + U_f c_{t-1} + b_f) \\i_t &= \sigma_g(W_i x_t + U_i c_{t-1} + b_i) \\o_t &= \sigma_g(W_o x_t + U_o c_{t-1} + b_o) \\c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + b_c) \\h_t &= o_t \circ \sigma_h(c_t)\end{aligned}$$

- x_t — входной вектор,
- h_t — выходной вектор,
- c_t — вектор состояний,
- W , U и b — матрицы параметров и вектор,
- f_t , i_t и o_t — векторы вентиляей,
 - f_t — вектор вентиля забывания, вес запоминания старой информации,
 - i_t — вектор входного вентиля, вес получения новой информации,
 - o_t — вектор выходного вентиля, кандидат на выход.

Рекуррентные нейрон с вентилями (Gated recurrent unit, GRU)



– Вентиль обновления: $z_t^j = \alpha(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1})^j$.

– Вентиль перезаписи: $r_t^j = J(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1})^j$.

– Кандидат на активацию: $\tilde{h}_t^j = \tanh(W_t + U(\mathbf{r}_t * \mathbf{h}_{t-1}))^j$,

$$h_{jt} = (1 - \frac{j_z}{j_{t-1}} + z_t^j \tilde{h}_t^j) h_{1i}$$

GRU

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)$$

- x_t : входной вектор
- h_t : выходной вектор
- z_t : вектор вентиля обновления
- r_t : вектор вентиля сброса
- W , U и b : матрицы параметров и вектор

