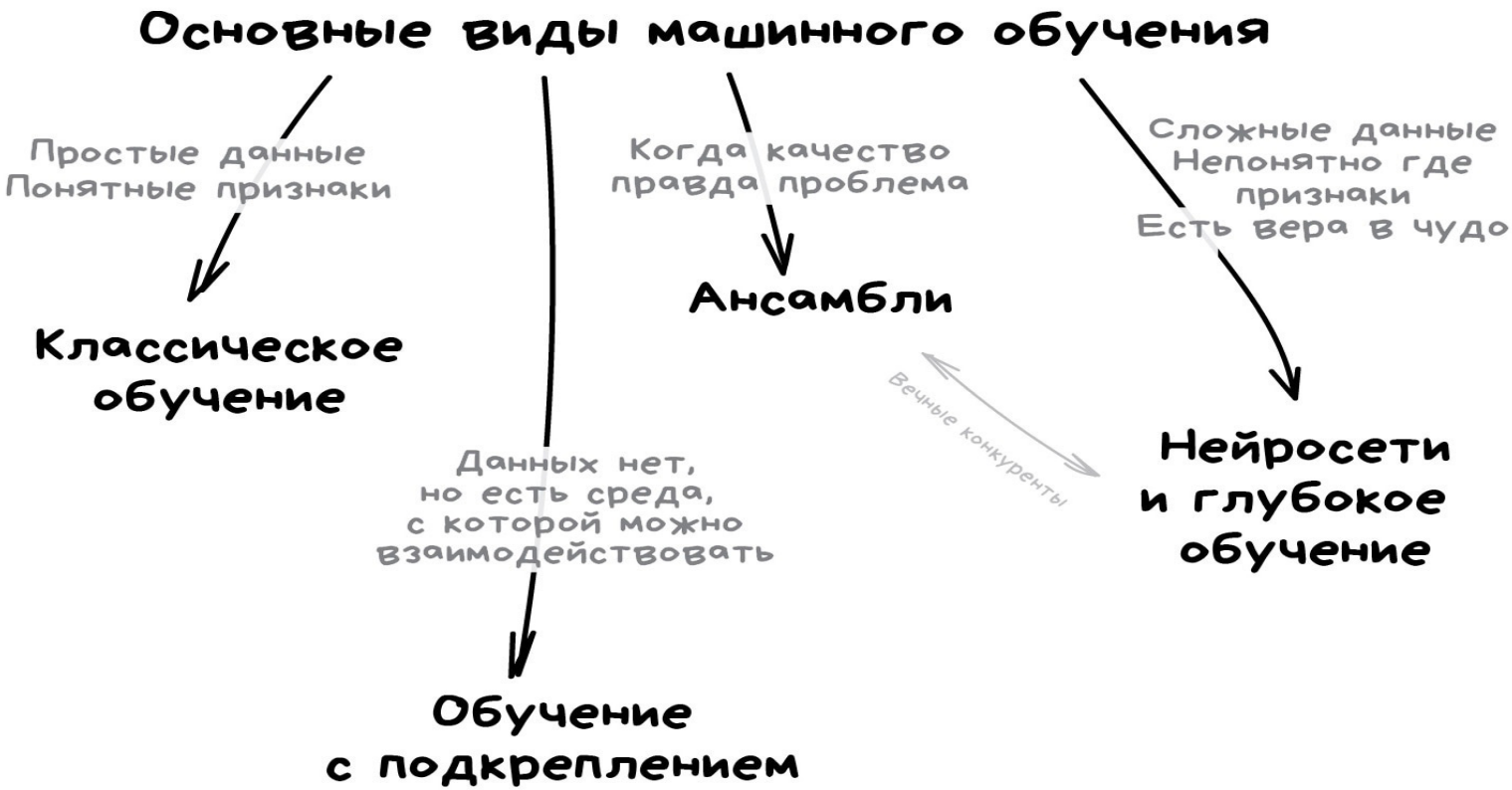
Deep Learning

Страшко Владислав
Сандул Михаил
Рукавишникова Анна

Санкт-Петербург
2019г.


Основные виды машинного обучения
Простые данные
Понятные признаки
Классическое обучение
Данных нет,
но есть среда,
с которой можно
взаимодействовать
Обучение
с подкреплением
Когда качество
правда проблема
Ансамбли
Вечные конкуренты
Сложные данные
Непонятно где
признаки
Есть вера в чудо
Нейросети
и глубокое
обучение

Классическое обучение: простые данные, понятные фиксированные признаки
Глубокое обучение: сложные данные, признаки метод «выучивает» сам

Основные представители:

- сверточные нейронные сети (CNN): анализ изображений, текстов, речи. Обобщаются на случаи данных с некоторой локальной структурой;
- рекуррентные нейронные сети (RNN): обработка последовательностей векторов.

Особенности:

- модель глубокого обучения собирается из большого количества блоков (сверточных, рекуррентных, полносвязных, ...);
- обучение на GPU (TPU)

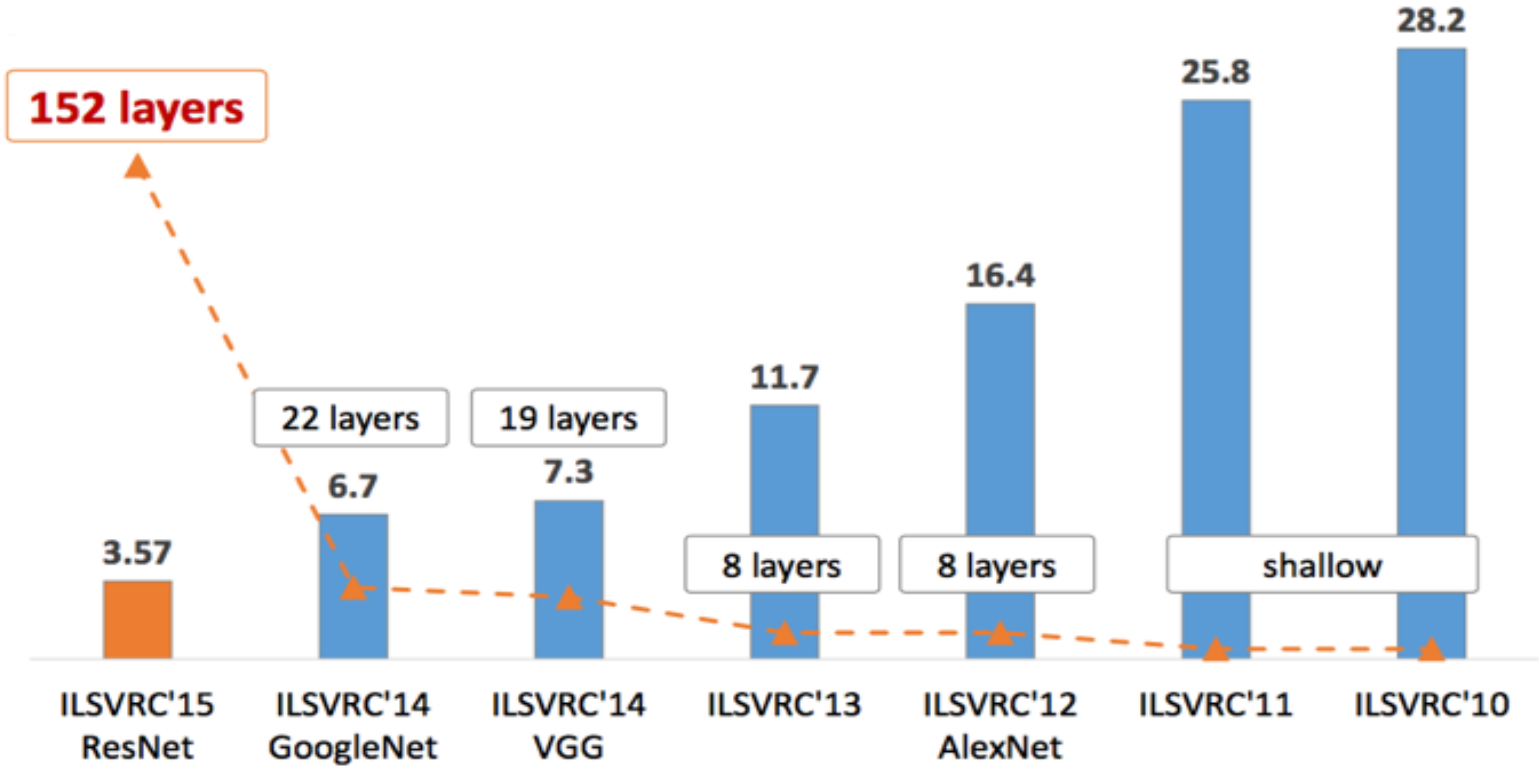


Рис.: ImageNet Classification top-5 error

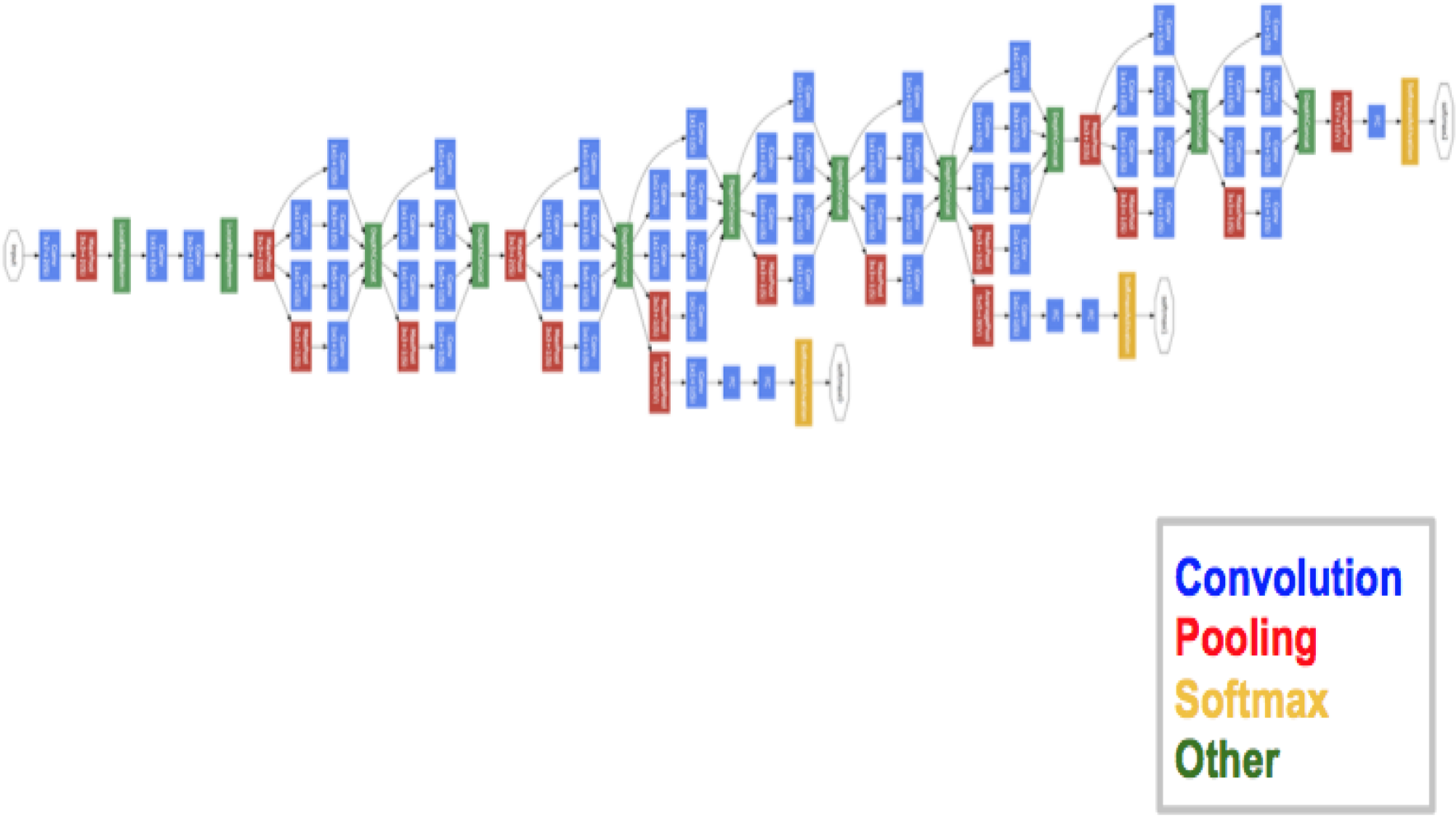


Рис.: Архитектура GoogleNet

Для задачи обработки последовательностей:

x_t — входной вектор в момент t;
s_t — вектор скрытого состояния в момент t;
o_t — выходной вектор

$$s_t = \sigma_s(Ux_t + Ws_{t-1})$$
$$o_t = \sigma_o(Vs_t)$$

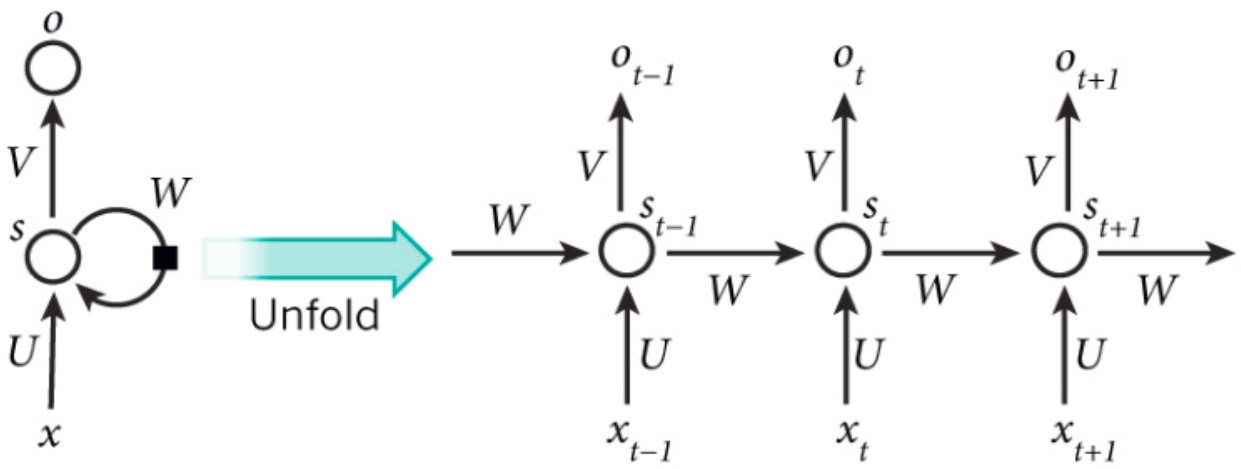


Рис.: Рекуррентная нейронная сеть

Обучение рекуррентной сети:

$$\sum_{t=0}^{T} \mathcal{L}_t(U, V, W) \to \min_{U,V,W},$$

где $\mathcal{L}_t(U, V, W) = \mathcal{L}(y_t(U, V, W))$ — потеря от предсказания y_t

Специальный вариант обратного распространения ошибок, Backpropagation Through Time (BPTT):

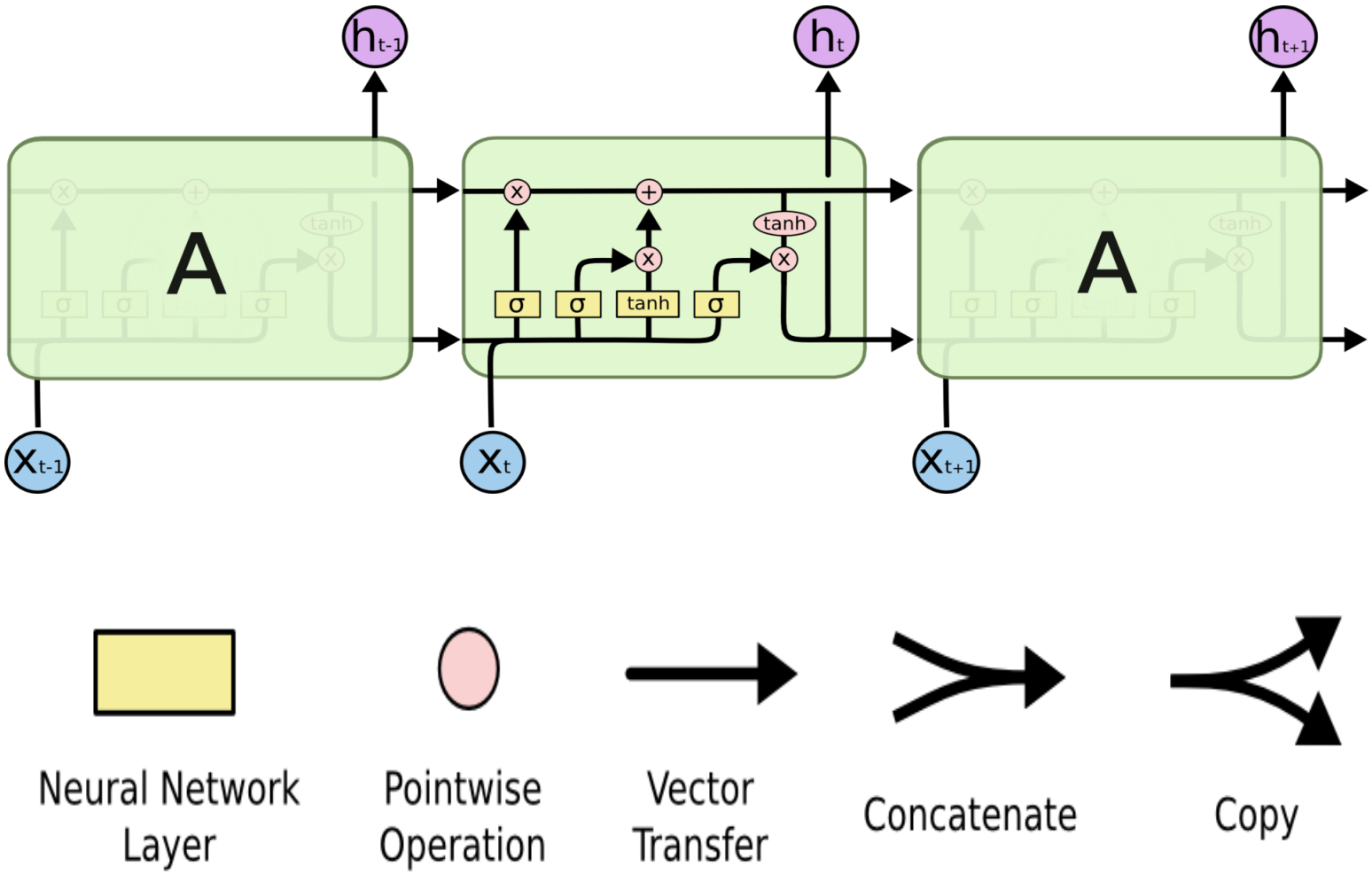
$$\frac{\partial \mathcal{L}_t}{\partial W} = \frac{\partial \mathcal{L}_t}{\partial o_t} \frac{\partial o_t}{\partial s_t} \sum_{k=0}^{t} \left(\prod_{i=k+1}^{t} \frac{\partial s_i}{\partial s_{i-1}} \right) \frac{\partial s_k}{\partial W}$$

Здесь есть 2 проблемы:

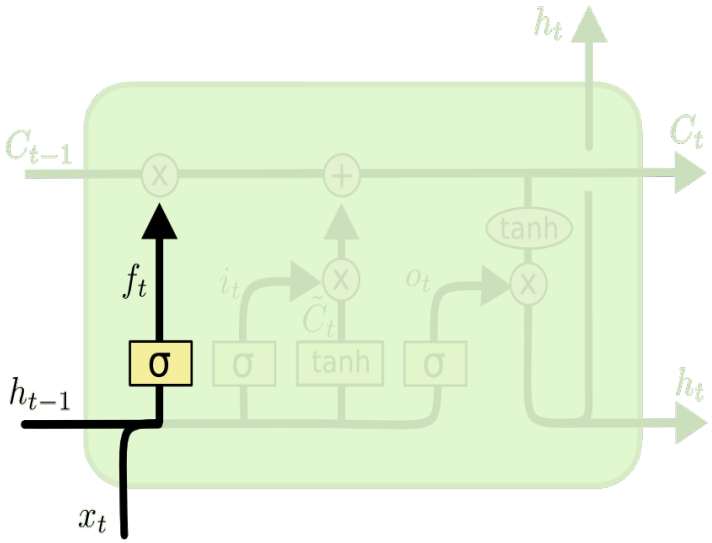
1. нам не нужно помнить всю историю — можем считать только несколько последних скрытых состояний
2. Затухание/взрыв градиентов, если $\frac{\partial s_i}{\partial s_{i-1}} \not\to 1$ — ограничить частную производную(ввести архитектуру, чтобы эта величина стремилась к 1)

LSTM (long short-term memory)

Мотивация LSTM: сеть должна долго помнить контекст, какой именно — сеть должна выучить сама. Поэтому вводится C_t — вектор состояния сети в момент t.

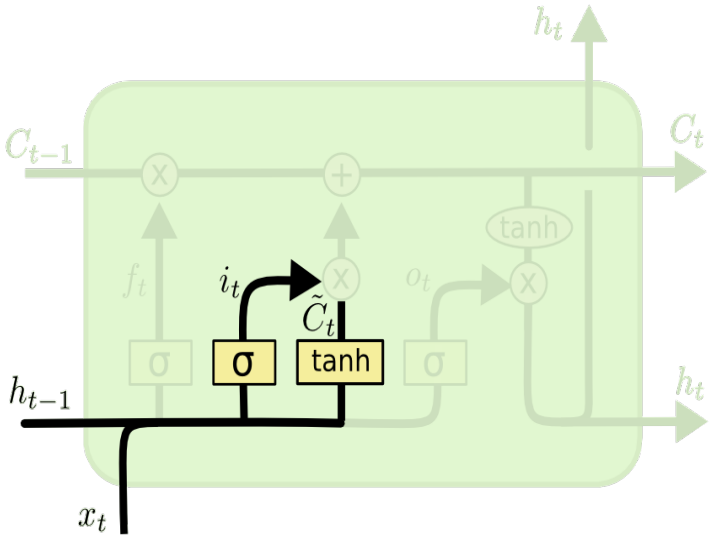


LSTM (long short-term memory)



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Фильтр забывания (forget gate) с параметрами W_f, b_f решает, какие координаты вектора состояния C_{t-1} надо запоминать.
Своего рода «классификатор».
Здесь σ — сигмоида.



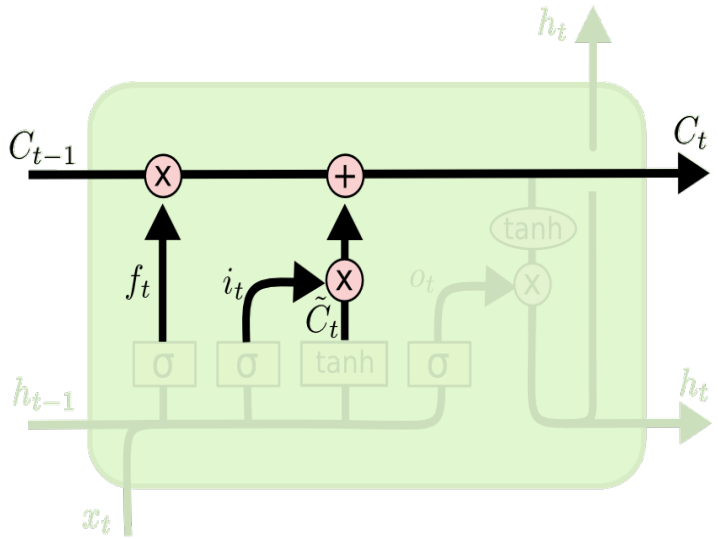
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Фильтр входных данных (input gate) с параметрами W_i, b_i решает, какие координаты вектора состояния надо обновить.

Модель нового состояния с параметрами W_C, b_C формирует вектор $\widetilde{C}_t$ значений-кандидатов нового состояния.

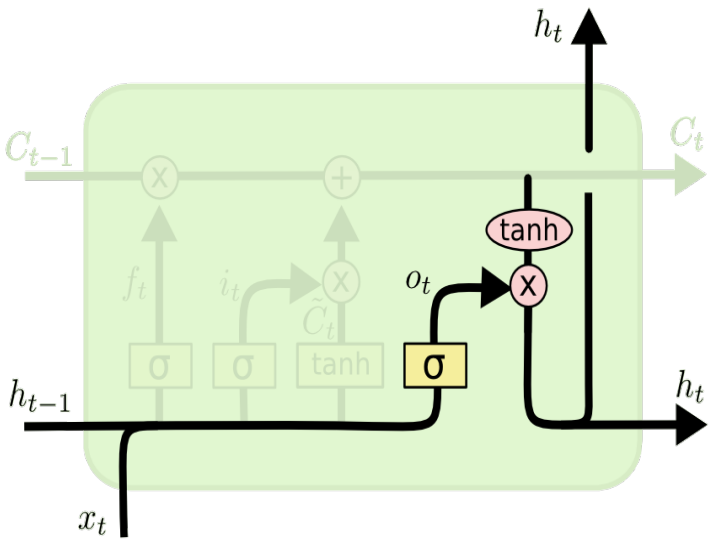
LSTM (long short-term memory)



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Новое состояние C_t формируется как смесь старого состояния C_{t-1} с фильтром f_t и вектора значений-кандидатов $\widetilde{C}_t$ с фильтром i_t.
Здесь операция покоординатного умножения.

Обучаемых параметров нет.



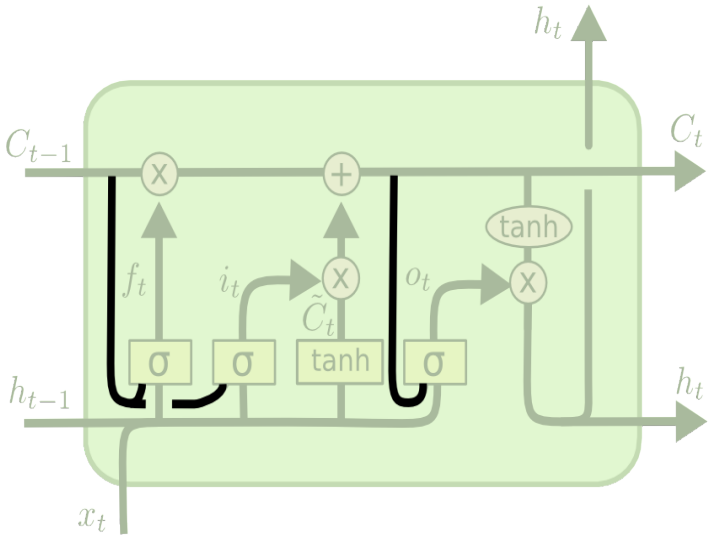
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Фильтр выходных данных (output gate) с параметрами W_o, b_o решает, какие координаты вектора состояния C_t надо учесть. Выходной сигнал h_t

формируется из вектора состояния C_t с помощью нелинейного преобразования th и фильтра o_t.

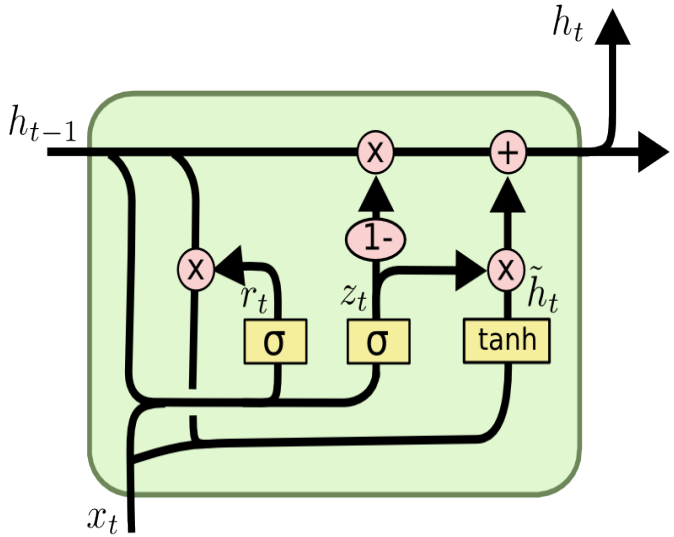
LSTM с «замочными скважинами»



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Здесь в каждом слое сети учитываются вектора состояния C_t и C_{t-1}.

LSTM: Gated Recurrent Unit (GRU)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Используется только состояние h_t, вектора C_t нет.
z_t вместо входного и забывающего фильтров.
Фильтр перезагрузки $\widetilde{h}_t$ решает, какую часть памяти нужно перенести дальше с прошлого шага.

Прогноз при помощи сетей прямого распространения:

- проходим окном длины L вдоль ряда, каждое окно — L-мерный элемент обучающей выборки
- в качестве входного слоя используем L последних наблюдений ряда, в качестве выходного — значение прогноза на 1 точку вперед

Рекуррентные сети:

- RNN — архитектура one-to-many
- предыдущий выход переиспользуется для предсказания на несколько точек. Частный случай — one-to-one

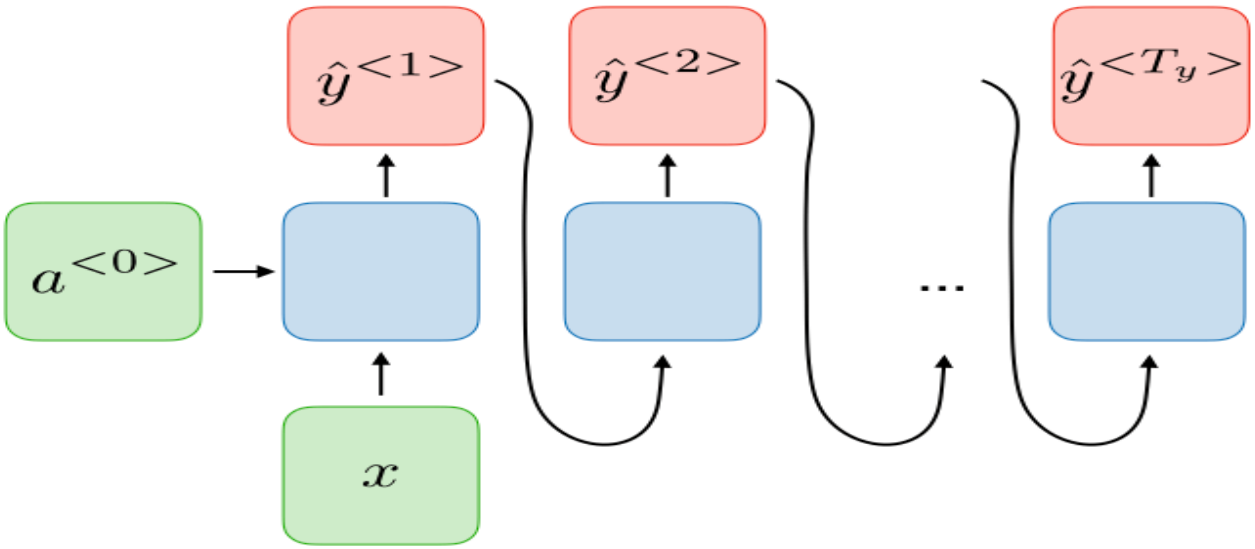


Рис.: RNN: One to many

Векторное представление объектов

Рекуррентная нейронная сеть принимает на вход вектора. Нужно каким-то образом вложить объекты в векторное пространство.

Word embeddings — семейство методов представления слов естественного языка в виде векторов

Желаемый результат вложения:

- Близкие вектора в смысле расстояния соответствуют семантически/синтаксически близким словам.
- Свойство аналогии слов: King − Man = Queen − Woman

word2vec

Модель: вероятность слова w быть в контексте слова u:

$$p(w|u) = \frac{\exp < x_w, x_u >}{\sum_v \exp < x_v, x_u >}$$

Оценка максимального правдоподобия:

$$\sum_{w,u \in W} n_{wu} \log p(w|u) \to \max_{x_w},$$

где n_{wu} — частота совместной встречаемости слов u и w