онлайн-курс

СПЕЦИАЛЬНЫЕ АРХИТЕКТУРЫ НЕЙРОННЫХ СЕТЕЙ

Обработка последовательностей

Общая идея

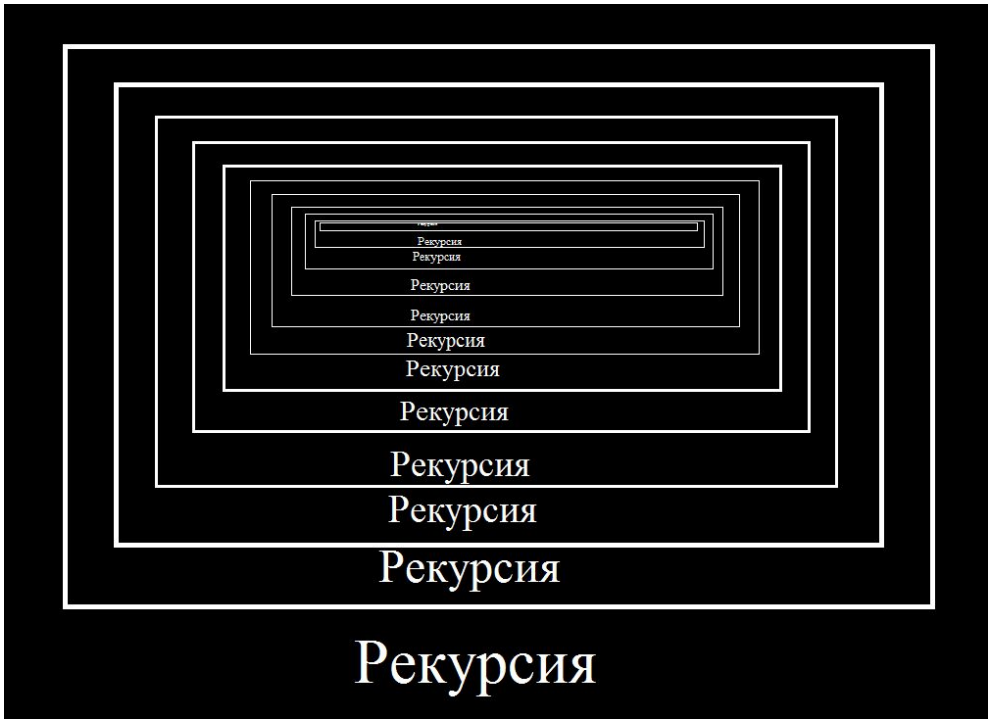
Что такое последовательность?

Сюда попадают разные типы данных: аудио, видео, тексты, временнЫе ряды (то есть зависимости разных величин от времени).

Важно, что в последовательности **очередное значение существенно зависит от предыдущих значений**.

Специальная архитектура

Для обработки последовательностей используются рекуррентные нейронные сети (РНС).



Важные особенности РНС

В «обычные» НС всегда подается один объект A, этот объект прогоняется через сеть, и мы смотрим на ответ. Результат работы для объекта A у такой НС полностью детерминированный, ответ не зависит от того, какие объекты были поданы в НС до A.

В РНС всё не так: для них существенно **важна история подачи объектов** на вход РНС. Результат работы РНС для объекта A существенно зависит от объектов, которые были поданы ранее.
А это очень важно при обработке последовательностей.

Осталось понять, как это реализовать с помощью математики.

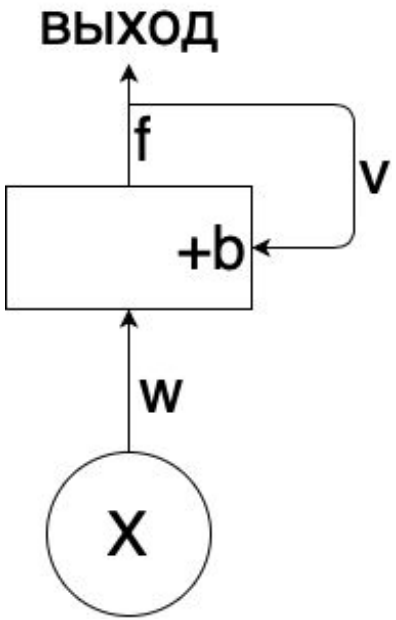
Идея!

Давайте на вход НС будем подавать **два объекта**: объект А (который мы хотим обработать) и **выходное значение НС, возникшее при обработке предыдущего объекта.**

какая метафора вам ближе?

Идея!

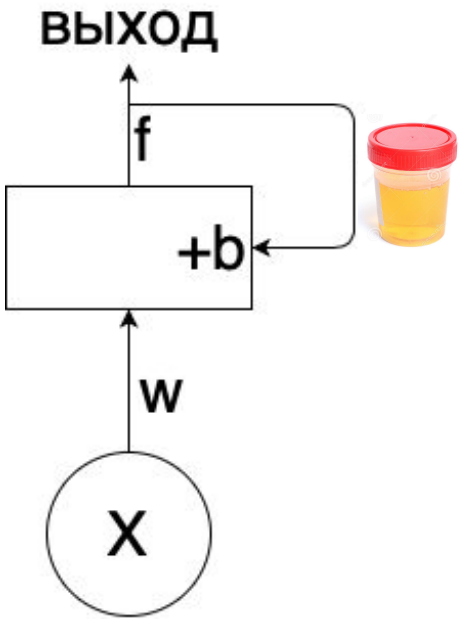
То есть появляется архитектура подобного типа:



Но есть проблема: как у такой НС вычислить функцию сети $F_{NN}(x)$?

Для тех, кто выбрал книгу (а не слоника)

То есть появляется архитектура подобного типа:



Но есть проблема: как у такой НС вычислить функцию сети $F_{NN}(x)$?

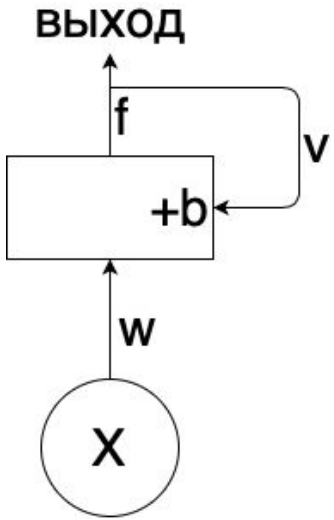
Как считать функцию сети?

$F_{NN}(x)=f(xw+b+v)$

$=f(xw+b+vf(xw+b+v))$

$=f(xw+b+vf(xw+b+vf(xw+b+v)))$

$=...$



Мы зациклились!

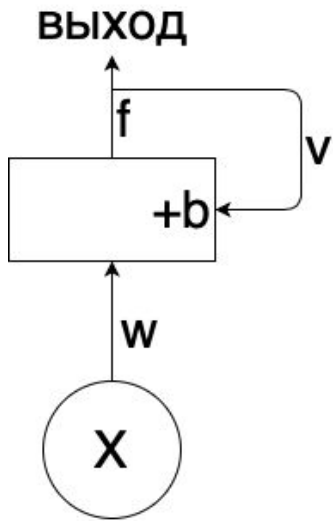
Да, в «общем виде» функция сети не выписывается.
Но оказывается, что можно обойтись и без этого.

Как поступить?

Используем историю входов сети

Зафиксируем **длину истории t** (то есть в работе мы будем использовать только последние t объектов, пропущенных через НС).

Пусть в нашу НС до объекта x были поданы объекты $x_1,\ldots,x_t$.
В этом случае рекурсия с прошлого слайда остановится через t шагов.



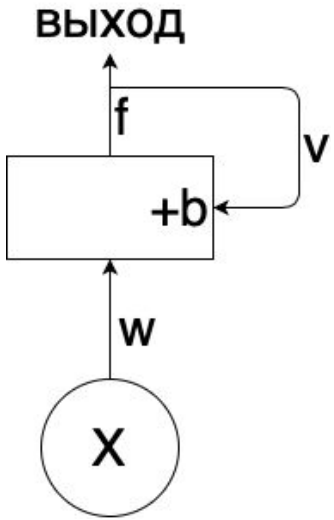
Как считать функцию сети?

Пусть t=3. Тогда функция сети для объекта x_3 равна:

$$F_{NN}(x_3)=f(x_3w+b+v)$$

$$=f(x_3w+b+vf(x_2w+b+v))$$

$$=f(x_3w+b+vf(x_2w+b+vf(x_1w+b+0)))$$

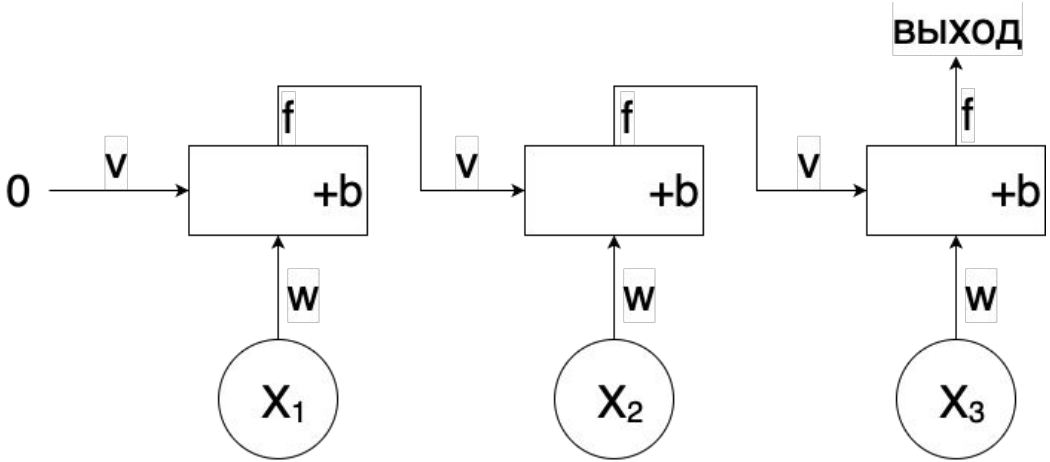


Мы предполагаем, что вклад более ранних объектов в ответ для объекта x_3 нулевой. Поэтому в нашей формуле и возникает 0.

Переменные в этой формуле нужно интерпретировать так:
x_3 - текущий объект,
x_2 - предыдущий объект,
x_1 - пред-предыдущий объект.

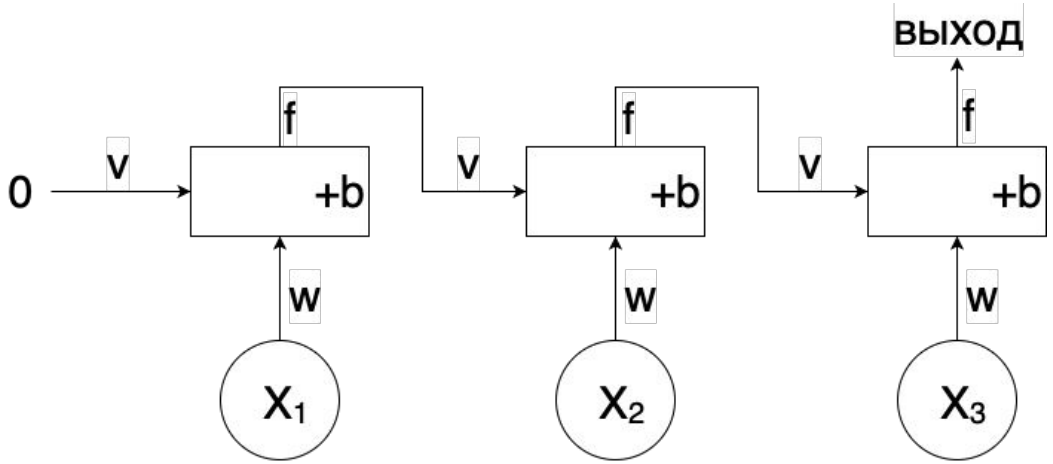
Развёртка сети

Если длина истории у нас равна t, то мы фактически создаем t копий НС и соединяем их цепочкой связей.



С функцией сети: $F_{NN}(x_3)=f(x_3w+b+vf(x_2w+b+vf(x_1w+b+0)))$

Развёртка сети



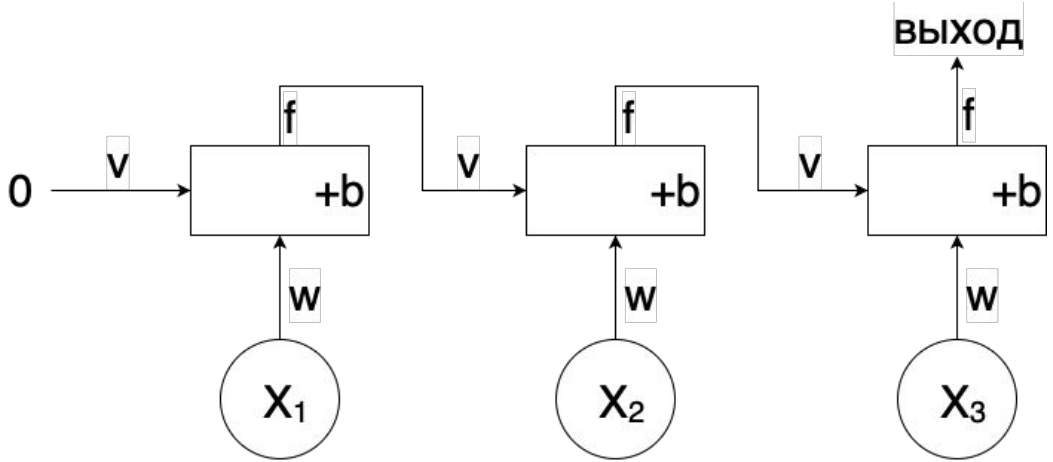
Для самого первого и второго объекта последовательности функция сети немножко другая:

$F_{NN}(x_2)=f(x_2w+b+vf(x_1w+b+0))$ $F_{NN}(x_1)=f(x_1w+b+0)$

Развёртка сети

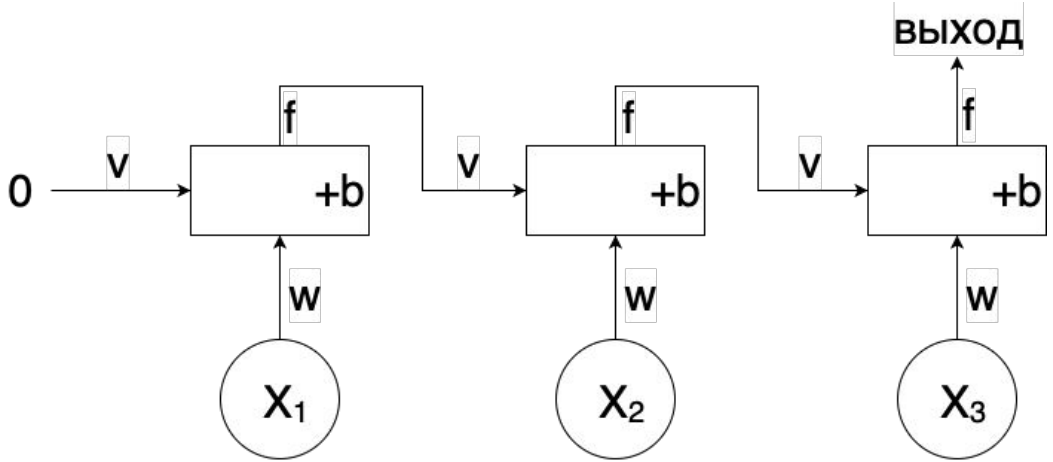
Важно! Во всех её частях **одни и те же веса и параметры**.

Входной объект x_i всегда домножается на **один и тот же** параметр w;
во всех нейронах прибавляется **одно и то же** смещение b;
всюду **одна и та же** функция активации f, **одно и то же** число v.



Длина истории

Длина истории (для НС на рисунке t=3) определяется эмпирически. То есть подбором.



Пример

Дана последовательность* : 1,1,2,3,5,8,...

Как вывести правило, по которому получается очередной член последовательности?

Допустим, что мы понятия не имеем о РНС, и пытаемся решить эту задачу с помощью «обычных» НС.

А обычной НС требуется ТВ в виде таблицы.
Как последовательность запихнуть в таблицу?

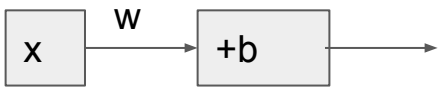
*18+: это последовательность Фибоначчи.

Первая попытка решения

Попытаемся составить таблицу с 2-мя столбцами X,Y по правилу: Y — это X-й член последовательности. Получаем:

X	Y
1	1
2	1
3	3
4	5
5	8

Выберем самую простую архитектуру НС:



и натренируем её обычными методами (как задачу регрессии).

Найдём такую зависимость…

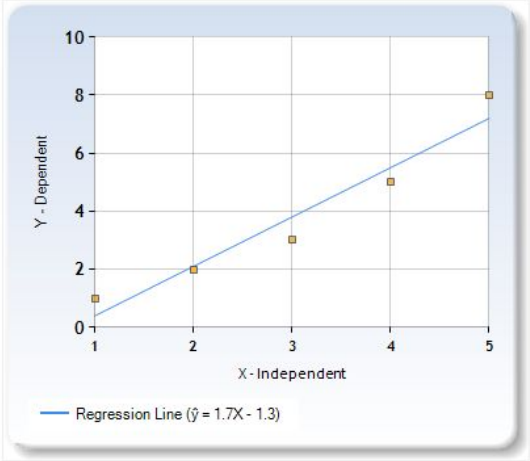
Первая попытка решения

Найдём такую зависимость **Y=1.7*X-1.3**

Она далека от истины, поскольку 6-й член последовательности в действительности равен 13, а не 1.7*6-1.3=8.9

А дальше расхождения станут ещё больше.

Попробуем по-другому сформировать ТВ…



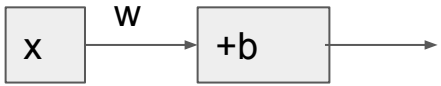
X	Y
1	1
2	2
3	3
4	5
5	8

Вторая попытка решения

Теперь составим таблицу с 2-мя столбцами X,Y по правилу:
Y следует за X в последовательности. Получаем:

X	Y
1	1
1	2
2	3
3	5
5	8

Выберем самую простую архитектуру НС:



и натренируем её обычными методами (как задачу регрессии).

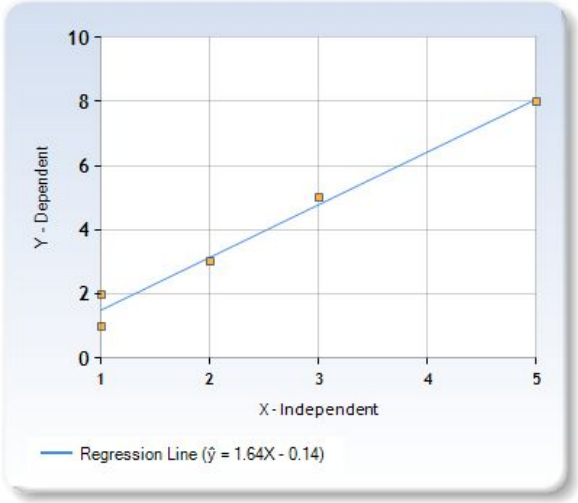
Найдём такую зависимость…

Вторая попытка решения

Найдём такую зависимость
Y=1.64*X-0.14

Она несколько получше, поскольку при X=8 она даёт 1.64*8-0.14=12.98 (правильный ответ 13). И даже для X=13 она попадает близко к 21.

Но дальше — хуже. Для X=610 она даёт 1.64*610-0.14=1000.26 (а не 987)*.



X	Y
1	1
1	2
2	3
3	5
5	8

*Это связано с золотым сечением и его приближением с помощью рациональных чисел.

Третья попытка решения

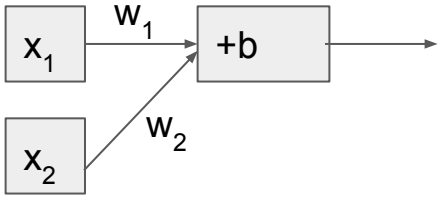
X_1	X_2	Y
1	1	2
1	2	3
2	3	5
3	5	8
5	8	13

Чтобы решить эту задачу с помощью обычных НС, нужно добавлять столбцы с **нецелевыми** признаками в таблицу, содержащие дополнительные сведения из истории.

Теперь составим таблицу с 3-мя столбцами X_1, X_2, Y по правилу: Y следует за парой X_1,X_2 в последовательности.

А вот тут-то истинная зависимость будет найдена:

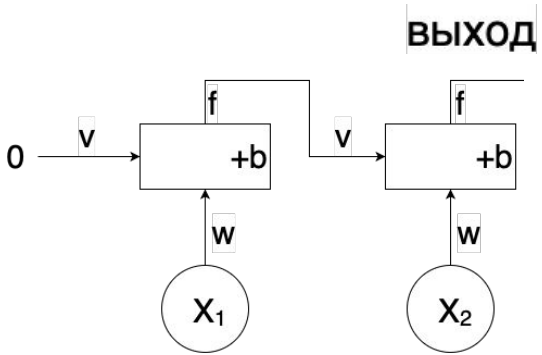
$Y=X_1+X_2$.



Вывод (вообще говоря, банальный): добавление истории в любом виде позволяет НС восстановить зависимость в последовательности.

Фактически это была РНС

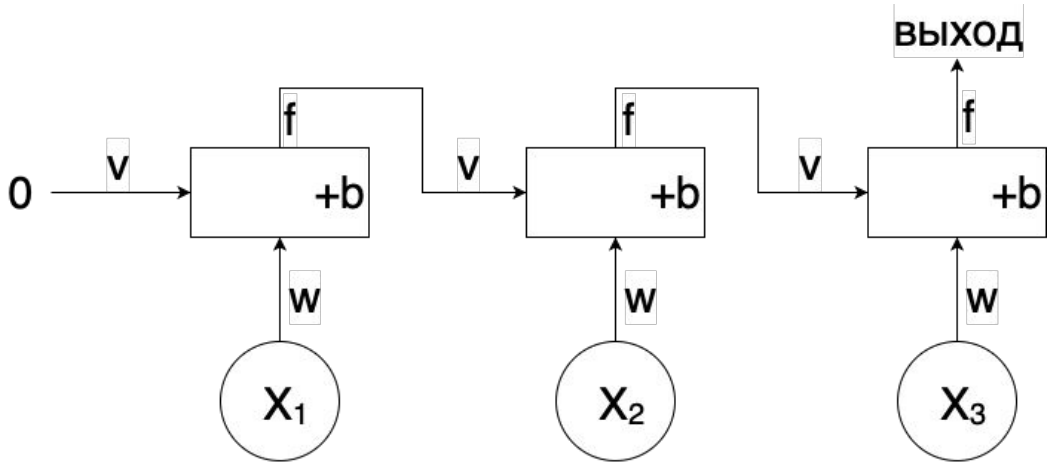
Фактически мы использовали РНС с длиной истории t=2
и архитектурой (функция активации тут тривиальная: f(x)=x):



для которой были найдены оптимальные веса: w=1, b=0, v=1.

Как же она тренируется?

Так же, как и обычная сеть. Рассмотрим тренировку этой РНС (для задачи **регрессии**):



Во-первых, вы выписываете функцию, которую вычисляет РНС:
$F_{NN}(x_3)=f(x_3w+b+vf(x_2w+b+vf(x_1w+b)))$.

Как же она тренируется?

Теперь берём последовательность ваших объектов: a_1,a_2,a_3,a_4,a_5, a_6 (их 6 шт.)

Поскольку входов у вашей РНС три, то задача заключается в предсказании очередного элемента последовательности по трём предыдущим. Последовательность развёртывается в таблицу:

Тройка	След. элемент	Что даёт РНС	Разница
a_1,a_2,a_3	a_4	$f(a_3w+b+vf(a_2w+b+vf(a_1w+b)))$	$f(a_3w+b+vf(a_2w+b+vf(a_1w+b)))-a_4$
a_2,a_3,a_4	a_5	$f(a_4w+b+vf(a_3w+b+vf(a_2w+b)))$	$f(a_4w+b+vf(a_3w+b+vf(a_2w+b)))-a_5$
a_3,a_4,a_5	a_6	$f(a_5w+b+vf(a_4w+b+vf(a_3w+b)))$	$f(a_5w+b+vf(a_4w+b+vf(a_3w+b)))-a_6$

Разница между истинным и фактическим предсказанием позволяет выписать функцию потерь:

$$L(w,b,v)=(f(a_3w+b+vf(a_2w+b+vf(a_1w+b)))-a_4)^2$$
$$+(f(a_4w+b+vf(a_3w+b+vf(a_2w+b)))-a_5)^2+ +(f(a_5w+b+vf(a_4w+b+vf(a_3w+b)))-a_6)^2,$$

которая минимизируется с помощью ГС, и мы находим оптимальные значения весов w,b,v.

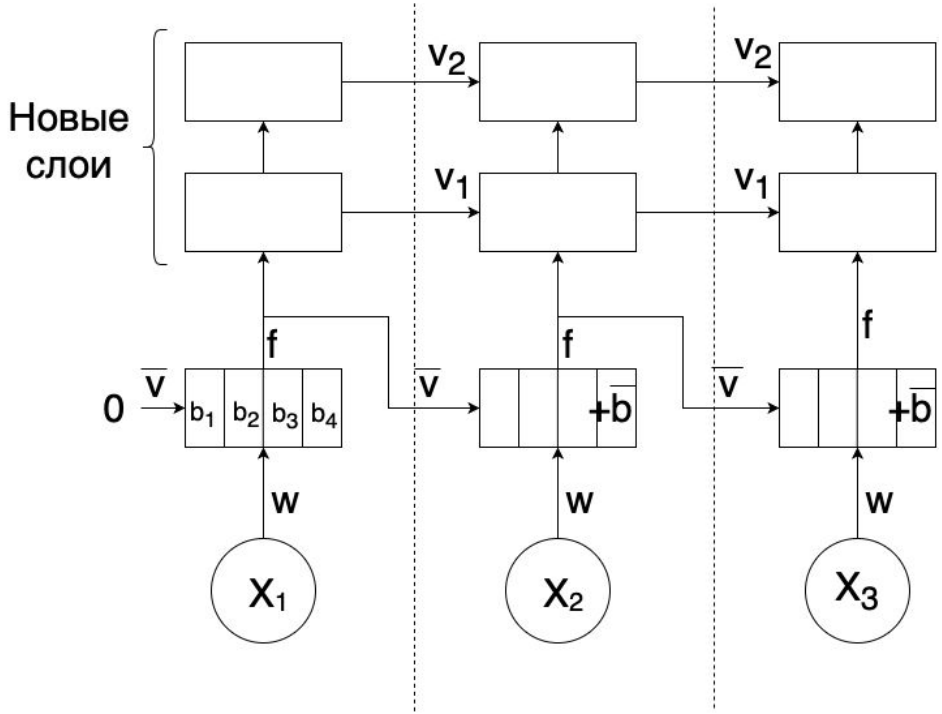
Тройка	След. элемент	Что даёт НС	Разница
a_1,a_2,a_3	a_4	$f(a_3w+b+vf(a_2w+b+vf(a_1w+b)))$	$f(a_3w+b+vf(a_2w+b+vf(a_1w+b)))-a_4$
a_2,a_3,a_4	a_5	$f(a_4w+b+vf(a_3w+b+vf(a_2w+b)))$	$f(a_4w+b+vf(a_3w+b+vf(a_2w+b)))-a_5$
a_3,a_4,a_5	a_6	$f(a_5w+b+vf(a_4w+b+vf(a_3w+b)))$	$f(a_5w+b+vf(a_4w+b+vf(a_3w+b)))-a_6$

Сложные архитектуры РНС

Рассмотренный выше простейший тип РНС был самым простым. Можно увеличивать число слоёв и длину истории.

Можно также вместо числа v использовать вектор v (это позволит передавать больше информации от предыдущих объектов).

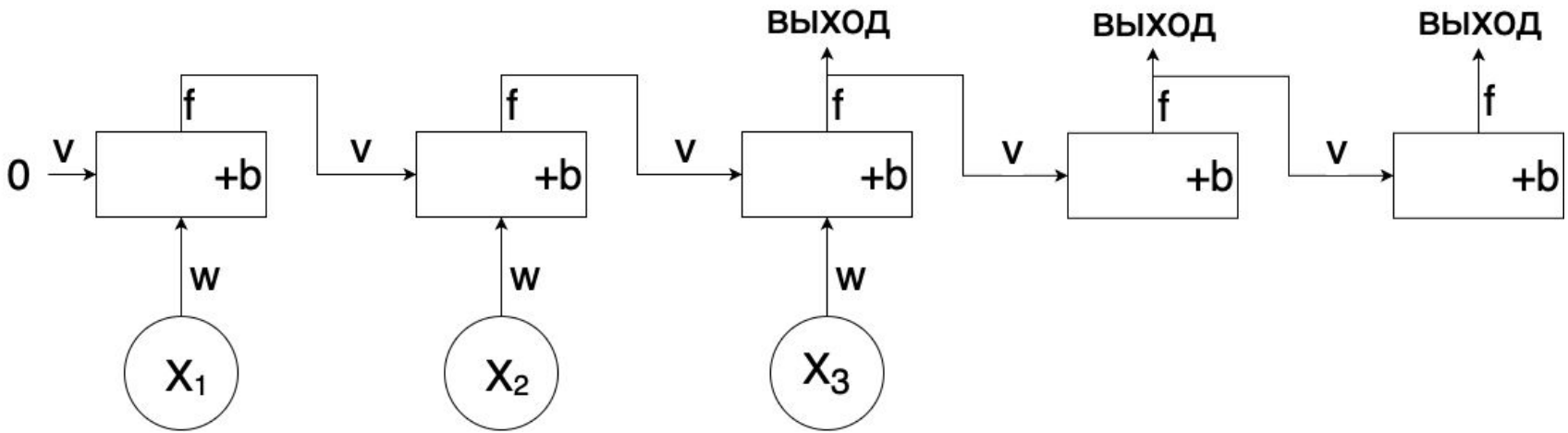
Например, на этом рисунке вектор v состоит из 4 координат, то есть от предыдущих объектов мы можем передать 4 числа.



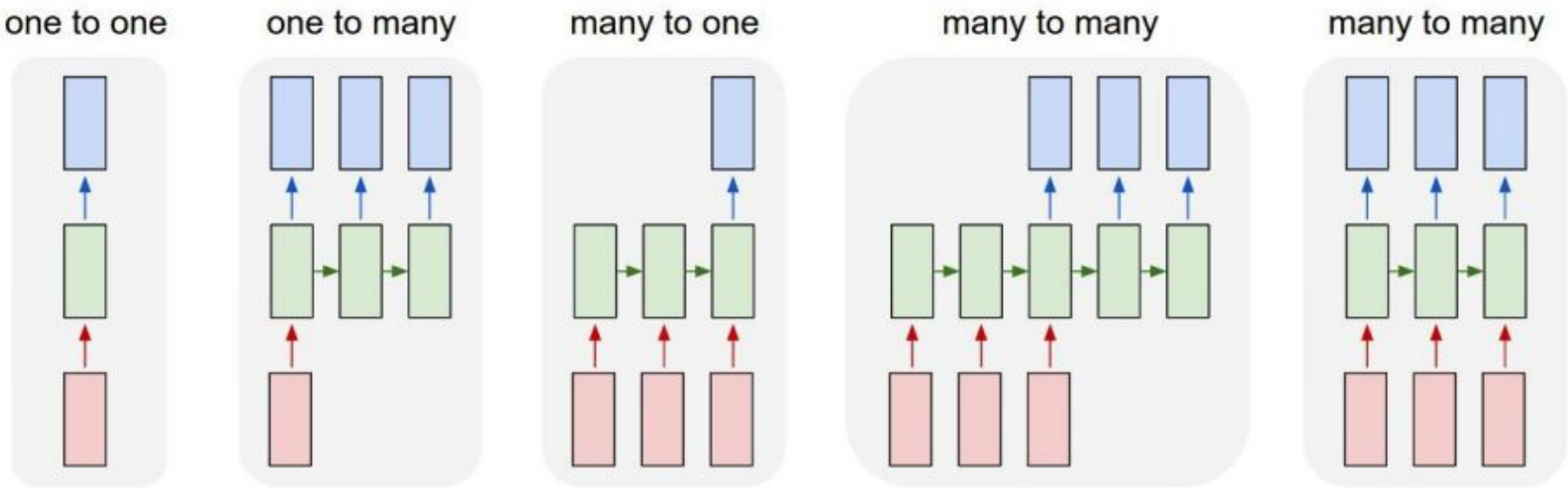
Ещё архитектура

Наконец можно сделать архитектуру, которая сначала «всасывает» в себя входные объекты, а потом генерирует последовательность выходов.

Это используется, например, при автоматическом переводе текста (сначала загружаем в РНС весь текст, а затем получаем перевод).



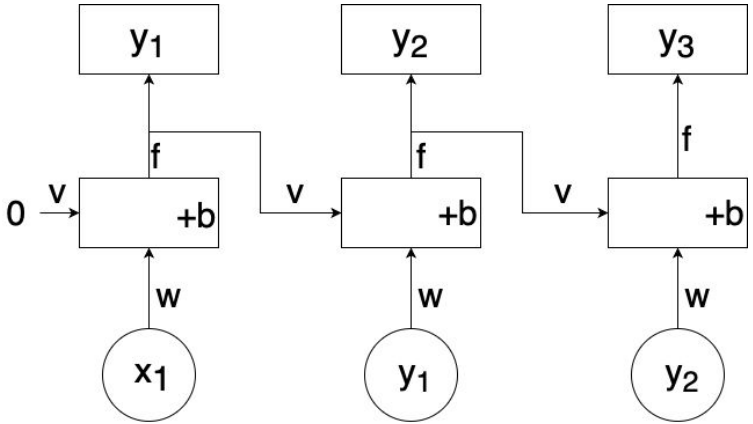
Тысячи их...



Генерация текста и его тональность

Будем генерировать тексты

Для этого используется РНС следующей архитектуры (для простоты длина истории t=3):



Важно: выход сверху совпадает со следующим ходом внизу.

x_1,y_1,y_2,y_3 — слова нашего языка.

Тренировочная выборка

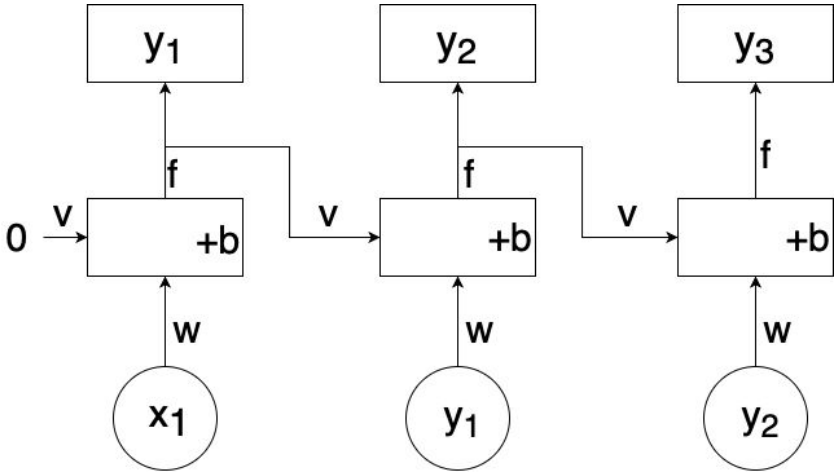
Тренировочная выборка для этой задачи формируется естественным образом: для тренировки таких сетей нужно загружать огромные массивы текстов.
Если вы хотите, чтобы сеть писала как Достоевский, в ТВ нужно загрузить все его труды.
После успешного обучения РНС будет генерировать тексты «в стиле Достоевского».

Подача ТВ в РНС

Начинаем перебирать весь корпус трудов Достоевского. Допустим, что мы дошли до фразы:

Это я убил тогда старуху-чиновницу и сестру её Лизавету топором, и ограбил

Исходя из архитектуры РНС, формируем тройки по правилу:



Тройка на вход РНС	Тройка на выход РНС

Получаем:

Это я убил тогда старуху-чиновницу и сестру её Лизавету топором, и ограбил

Вход в РНС	выход из РНС
это я убил	я убил тогда
я убил тогда	убил тогда старуху
убил тогда старуху	тогда старуху чиновницу
тогда старуху чиновницу	старуху чиновницу и
старуху чиновницу и	чиновницу и сестру
чиновницу и сестру	и сестру её

и так далее...

Оцифровка ТВ

Сейчас у нас ТВ состоит из групп слов. Их нужно превратить в векторы слов.

Спасибо **технологиям типа word2vec**. После этого у нас будет ТВ типа:

Тройка на вход РНС	Тройка на выход РНС
три вектора V, V’, V”	три вектора V’, V”, V””

По этой строке составляем слагаемое для функции потерь (самый простой способ):

$d(V,V')+d(V',V'')+d(V'',V''')$, где d — расстояние между векторами.

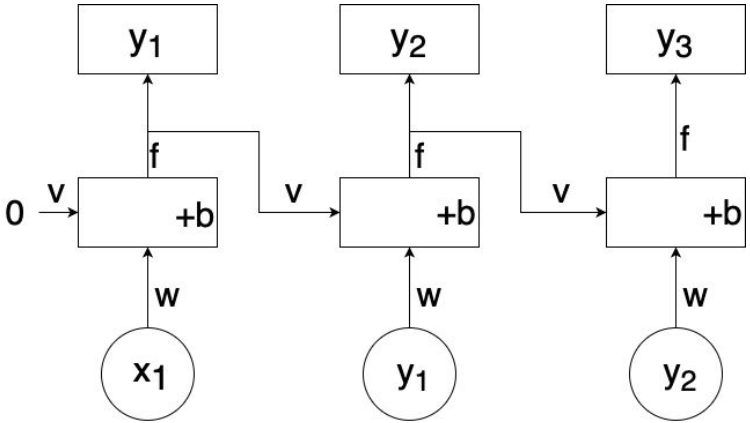
Суммируем это по всем строкам ТВ, получаем функцию потерь.

Обучение

Минимизируем функцию потерь, находим оптимальные значения весов РНС.

Как теперь генерировать текст?

Придумываем какое-нибудь первое слово текста.
Например: x_1=«старуха».



Обучение

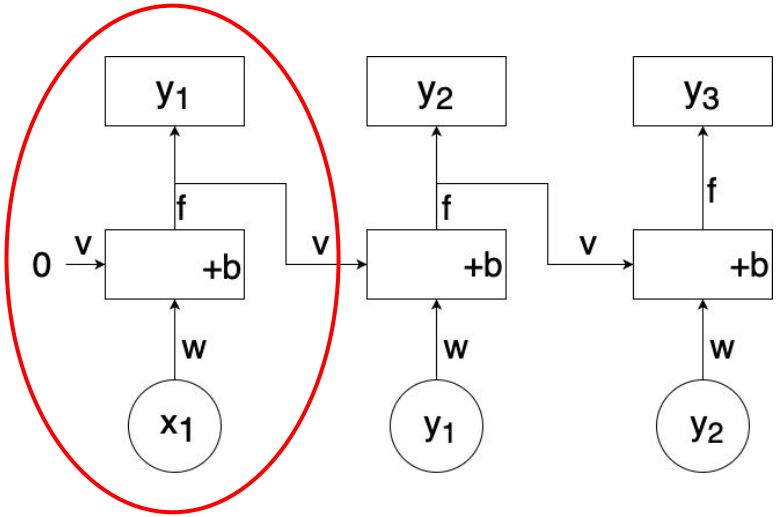
Минимизируем функцию потерь, находим оптимальные значения весов РНС.
Как теперь генерировать текст?

Придумываем какое-нибудь первое слово текста. Например: x_1=«старуха».

Пропускаем его только эту часть РНС:

Сеть выдает новое слово текста. Например: y_1=«процентщица».

Далее процедура повторяется: подаём на вход слово y_1=«процентщица» и получаем новое слово.



Результат

Ну и если вам повезёт,
то получится текст
«в духе Достоевского».

Совпадение сюжетных линий,
естественно, не гарантируется.

Игры с тональностью текста

Можно ли генерировать тексты
с определённой тональностью
(позитивной, негативной и нейтральной)?

Оказывается, можно!

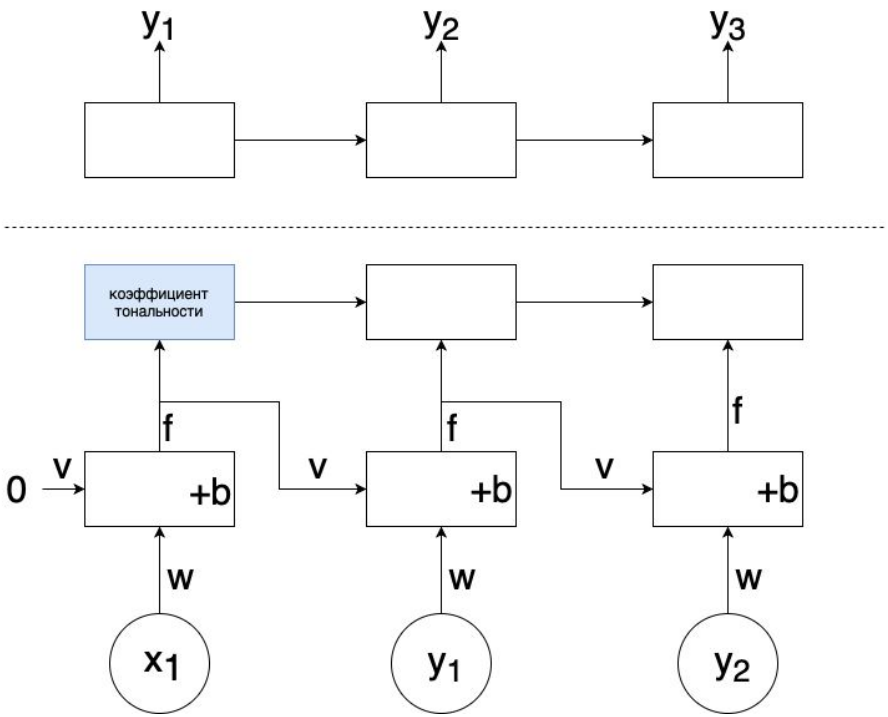
Что для этого нужно:

Во-первых, вы должны научиться
качественно генерировать сами тексты.

А дальше: ищите и обрящете…

… нужный нейрон в РНС, который отвечает за тональность всего текста.

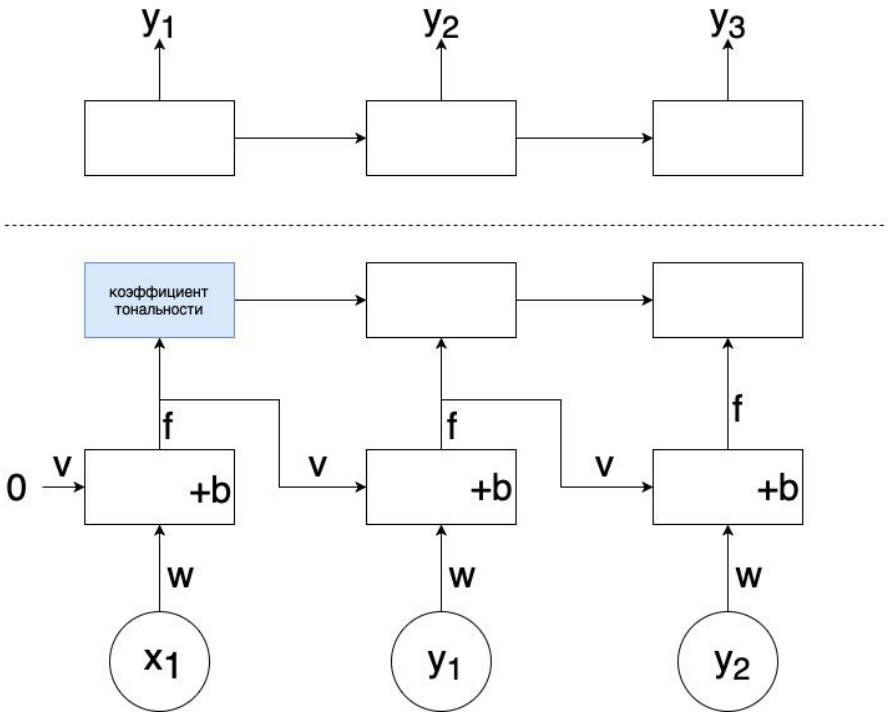
У этого нейрона выходные значения должны коррелировать с тональностью.



А дальше: ищите и обрящете…

Однажды группа специалистов создала глубокую РНС, натренировали НС и начали анализировать, какие значения выдаёт каждый из нейронов этой сети.

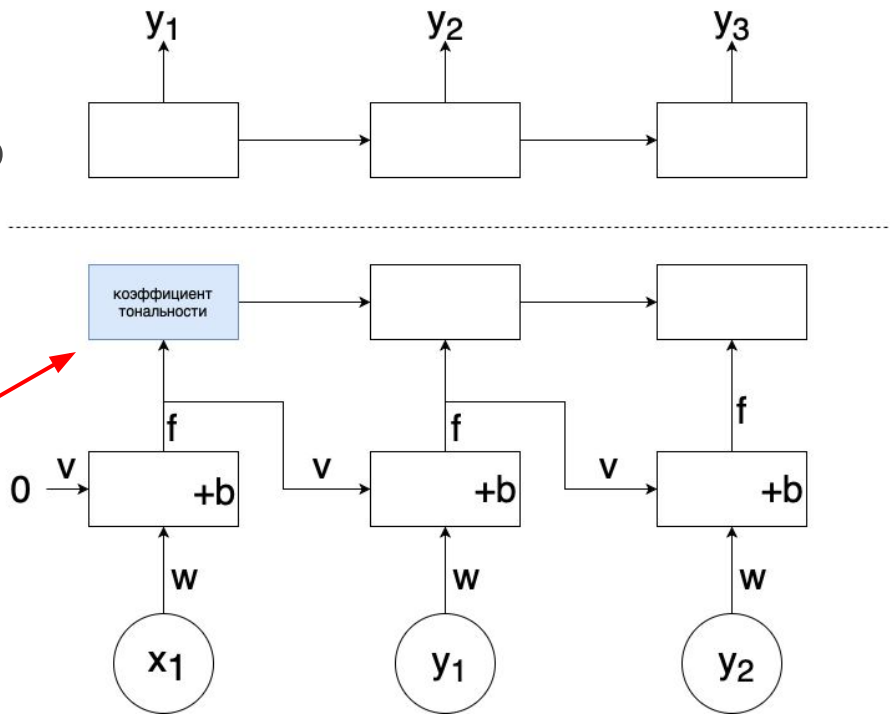
Натренировали сеть на базе ответов сайта типа КиноПоиска, где была явно указана тональность.



А дальше: ищите и обрящете…

Далее ей стали отдельно подавать
на вход тексты двух видов:
тексты с отрицательной тональностью
и тексты с положительной.

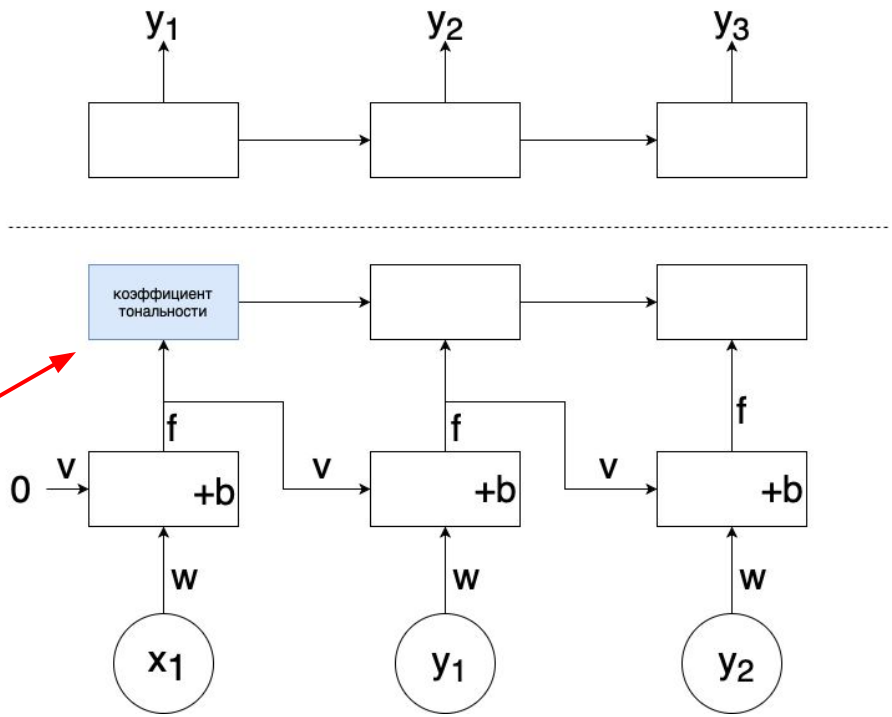
Оказалось, что в сети существует
нейрон, значение которого
определяет тональность,
т.е. значения, которые он выдаёт,
существенно различались в
зависимости от тональности.



А дальше: ищите и обрящете…

Итак, этот нейрон для текстов с положительной тональностью выдавал положительные значения, а для текстов с отрицательной тональностью он выдавал отрицательные значения.

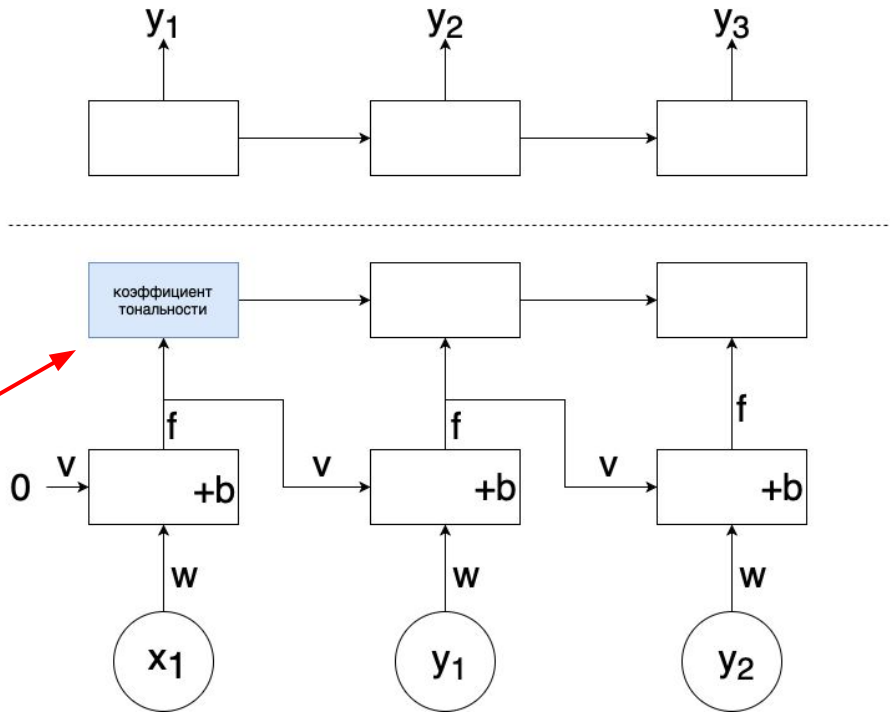
Как можно это использовать?
Если насильно заставлять этот нейрон выдавать только положительные значения, то вся РНС будет генерировать только позитивные тексты.



А дальше: ищите и обрящете…

Более того, можно менять тональность текста в процессе генерации.
Допустим, что первое слово было сгенерировано: «Дураки».
Ясно что, текст намечается быть с негативной тональностью.
Но мы теперь вмешиваемся в работу этого нейрона, и текст продолжается:

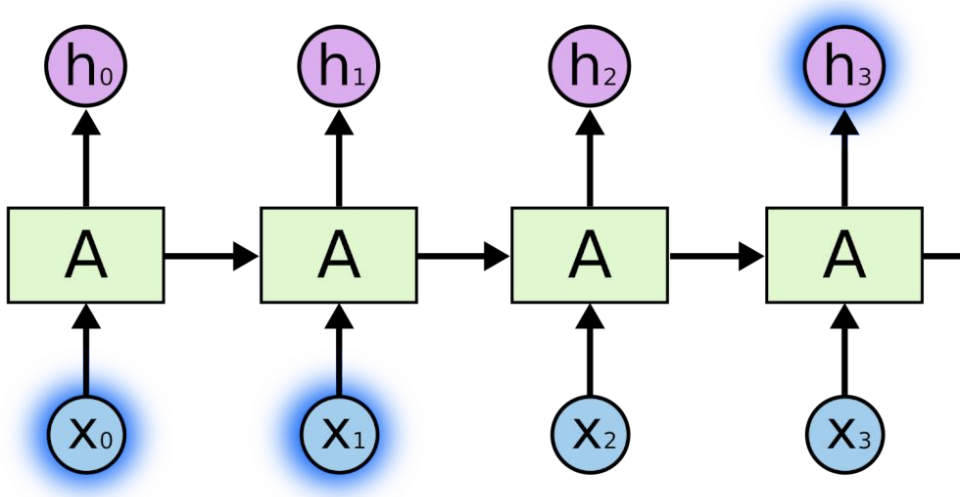
Дураки те, кто не понимает этот фильм. Фильм потрясающий.



LSTM-ячейки

Долгая история...

При генерации сложных объектов (тексты, музыка) очевидно, что длина истории у РНС должна быть большой. Поскольку очередной элемент последовательности может зависеть от очень далёких предыдущих элементов.



Пример с долгой историей: какое слово нужно сгенерировать?

Шёл по улице отряд —
сорок мальчиков
подряд:
раз,
два,
три,
четыре
и четырежды
четыре,
и четыре
на четыре,
и ещё потом…

Человек дополнит стих на основе ритма, а ИИ должен обратиться к истории и **фактически выполнить вычитание.**

Это единственный способ для ИИ закончить стих. Если доступа к далекой истории нет, то ИИ может основываться только на статистических (частотных) соображениях: «Что-то здесь мальчиков многовато. Для выправления процентного соотношения нужно добавить девочек. Поэтому закончим так…»

Пример с долгой историей: какое слово нужно сгенерировать?

Шёл по улице отряд —
сорок мальчиков
подряд:
раз,
два,
три,
четыре
и четырежды
четыре,
и четыре
на четыре,
и ещё потом **Наташа**.

И это ещё самая безобидная концовка.

Какую информацию нужно тащить за собой?

Должен быть механизм, позволяющий тащить информацию через сколь угодно большое количество слоев РНС.

При генерации текстов там обязательно будут слова (например, имена действующих лиц), которые зафиксированы в начале текста и потом регулярно упоминаются.

Кстати, **топор** у Достоевского упоминается только в самом начале романа и потом только в самом последнем предложении:

Это я убил тогда старуху-чиновницу и сестру её Лизавету топором, и ограбил

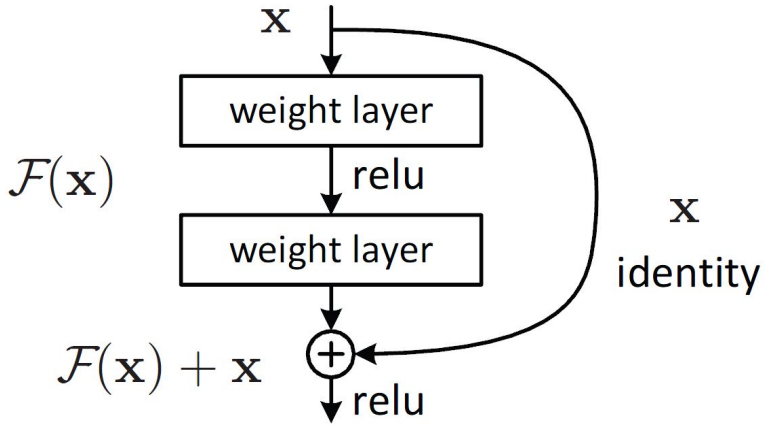
В общем, длинная история важна, но…

… возникают проблемы (как и в любой глубокой НС).

Это **затухание градиента**!
То есть чисто математически сложно протаскивать информацию через много слоёв.

Конечно, есть методы борьбы с затуханием градиента.

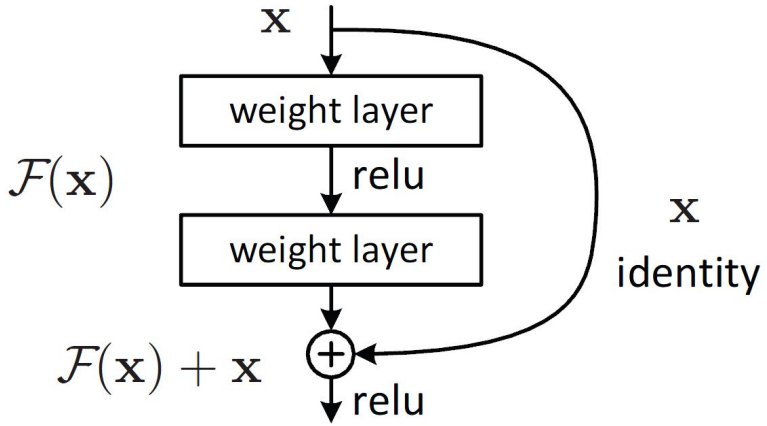
Например, идея из архитектуры СНС ResNet:



Архитектура ResNet...

... позволяет проталкивать информацию без искажений через сколь угодно большое количество слоев CHC.

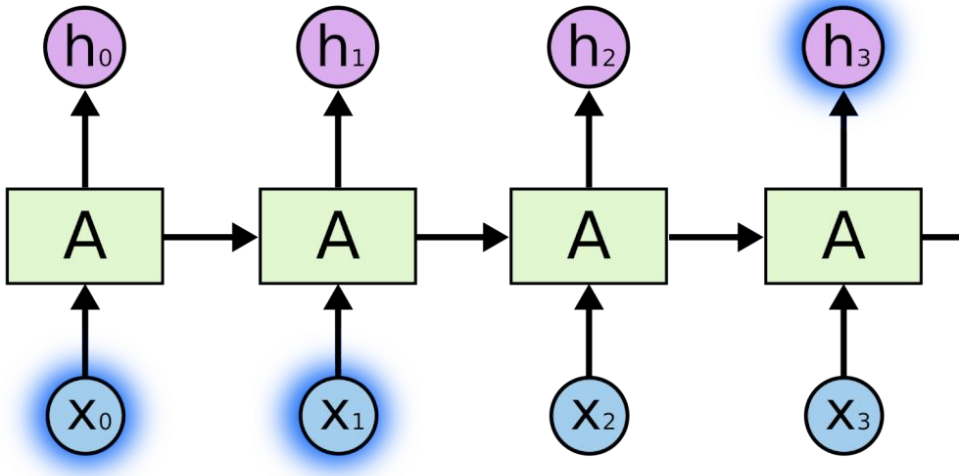
Но в случае с PHC возникает проблема.



Слишком много информации нужно тащить

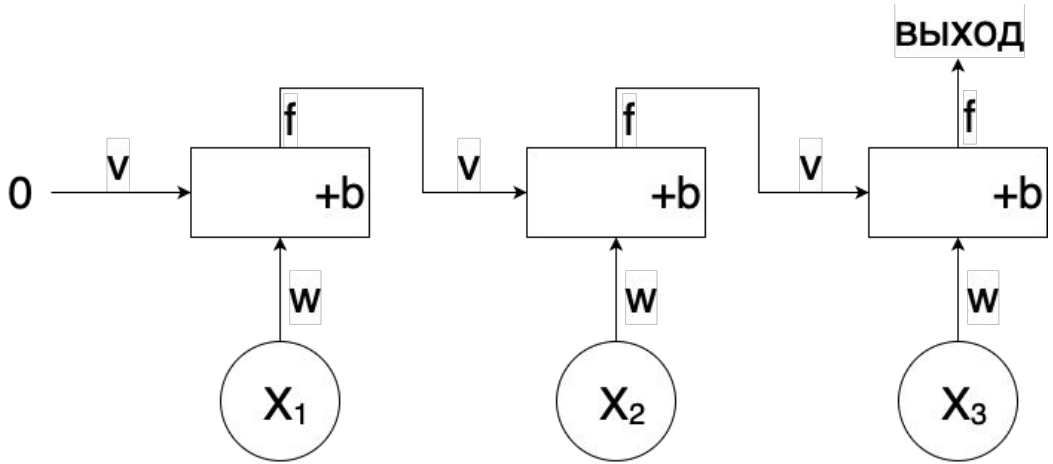
Дело в том, что у РНС много входов (количество входов равно длине истории).

Получается, что к последнему слою (см. схему) должна прийти информация от первых трёх слоев.



Дополнительная проблема

Слои РНС связаны с помощью **одного и того же веса** v

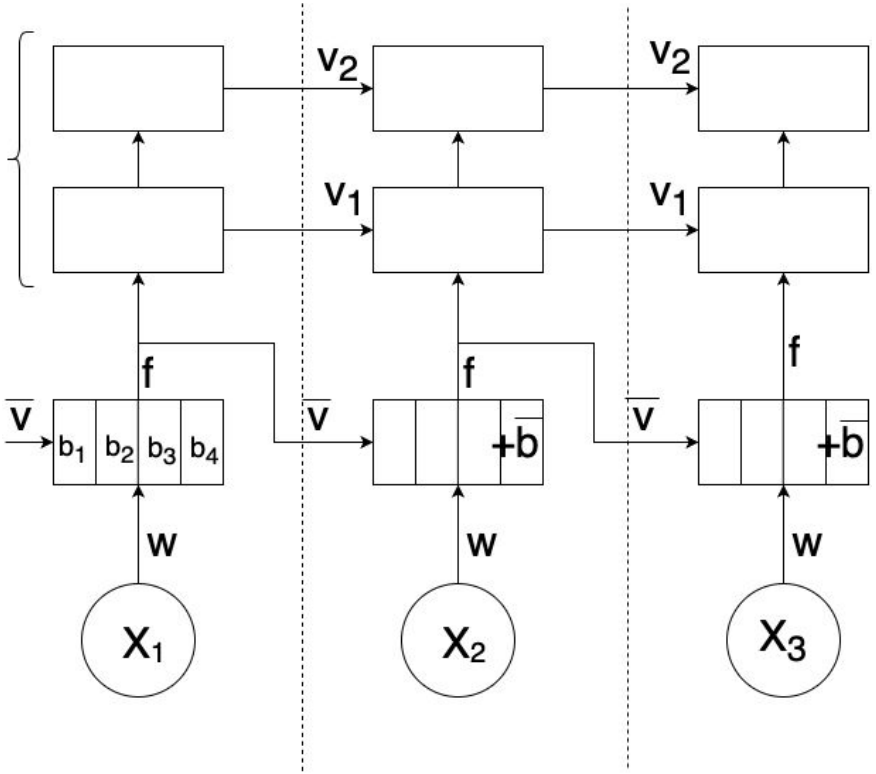


Дополнительная проблема

Даже если каждый слой РНС глубокий, всё равно между ними передаются одни и те же веса.

Это требование ускоряет тренировку РНС и для большинства задач вполне оправдано.

Но для нетривиальных проблем анализа (генерации) данных такое ограничение может стать проблемой.



Знакомая ситуация:

…информации много, а канал передачи не резиновый.

Решение стандартное: **архивация**.

А что подразумевает архивация?

- к информации с предыдущих слоёв добавляется информация с текущего слоя;
- выделение наиболее важных частей информации (их передаём дальше);

а также:

- удаление (забывание) ненужной информации, которая дальше не передаётся.

LSTM — это фактически архиватор,

осуществляющий работу по:

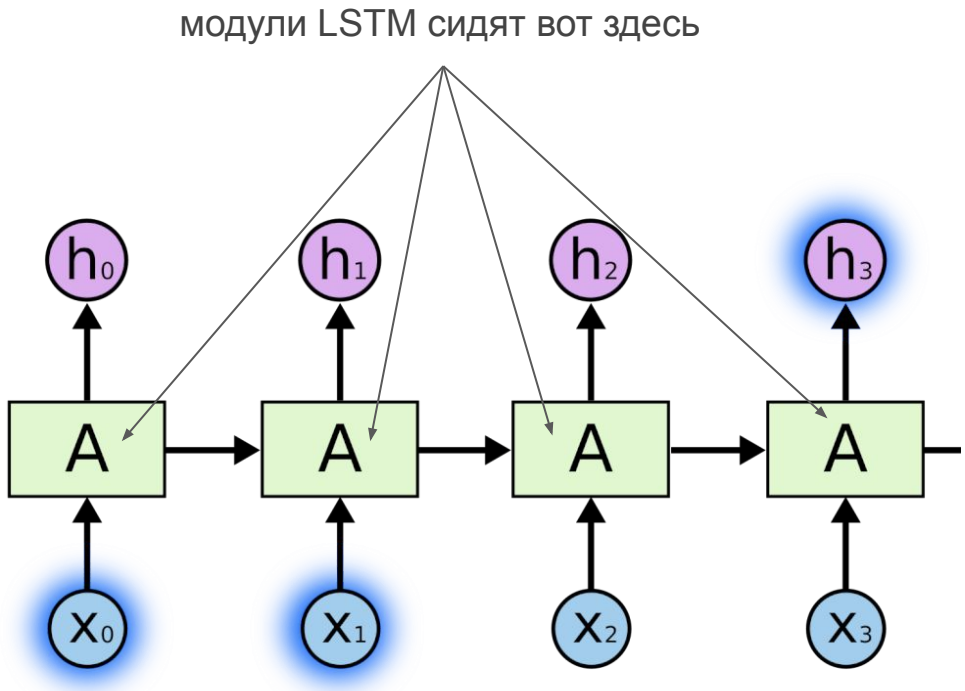
- соединению информации с прошлых слоёв и текущего слоя;
- забыванию ненужной информации;
- передаче релевантной информации дальше.

LSTM=long short-term memory.

LSTM **нужно тренировать** по ТВ.
У неё есть свои веса для тренировки.

LSTM — имплантируется в каждый слой РНС

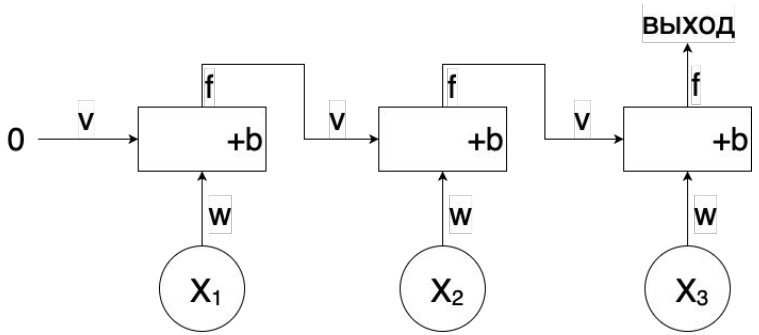
И у каждого модуля LSTM свои веса!



Из чего состоит LSTM-блок? Было:

Для простоты: далее все РНС работают с **данными размерности 1x1**. То есть на вход РСН подаются одиночные числа, и между слоями РНС передаются тоже числа.

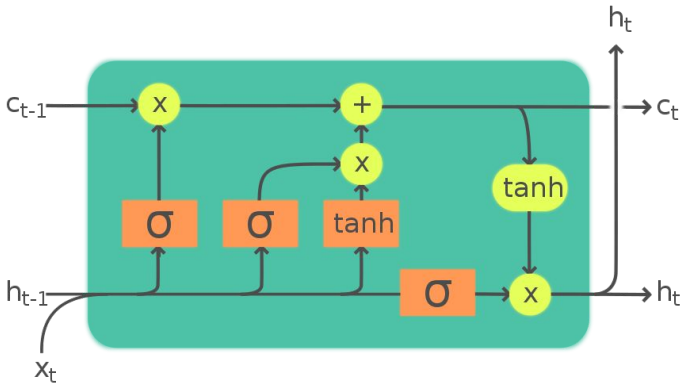
В стандартной архитектуре РНС с предыдущего слоя передается результат вычислений, умноженный на вес v.



Стало:

В РНС, состоящей из LSTM-блоков, между слоями с номерами t и t+1 передаются значения h_t, c_t.

Перед первым слоем РНС логично предположить, что $h_0=c_0=0$.



Их смысл:

h_t — краткосрочная память LSTM
(это полный аналог веса **v** с прошлых слайдов)

c_t — долгосрочная память LSTM

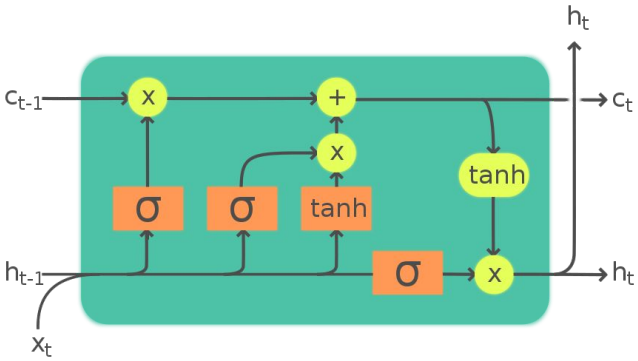
Из чего состоит LSTM-блок?

Мы видим, что всю РНС пронизывает «магистральная связь», по которой значения c_t из далёких ячеек могут быть без изменения переданы дальше.

Это будет при условии, что LSTM-блок не захочет изменять c_t.

Степень изменения значения c_t внутри очередного блока LSTM определяется в процессе тренировки РНС.

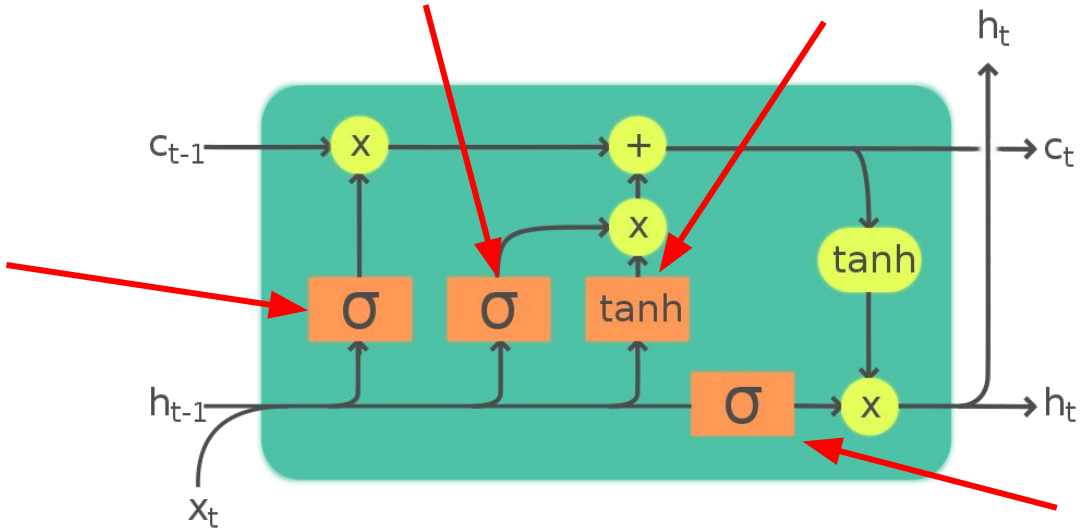
А теперь поподробнее о преобразованиях…



Из чего состоит LSTM-блок?

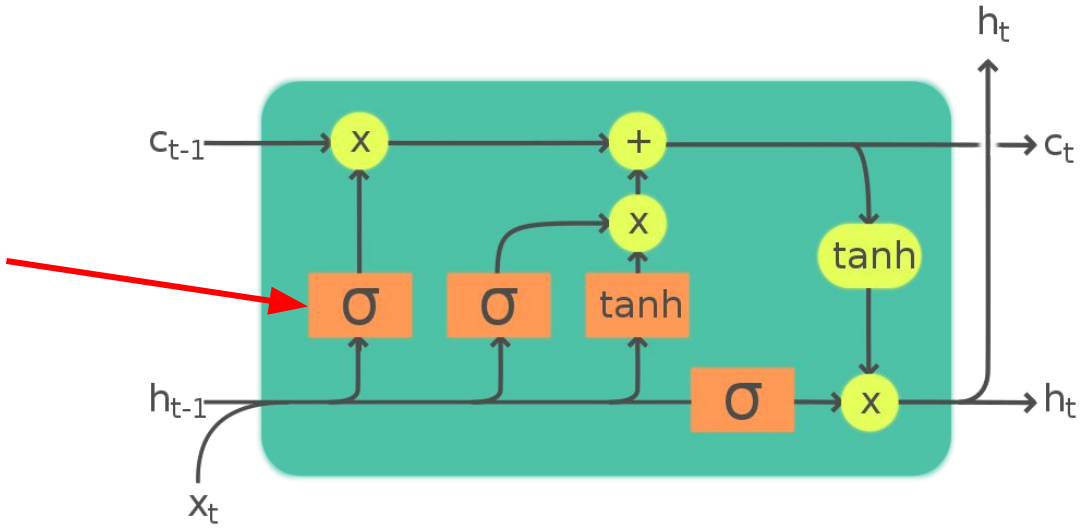
Мы изучим лишь архитектуру LSTM-блока (их много!).

Сначала рассмотрим, основные узлы (вентили, гейты) LSTM-блока.



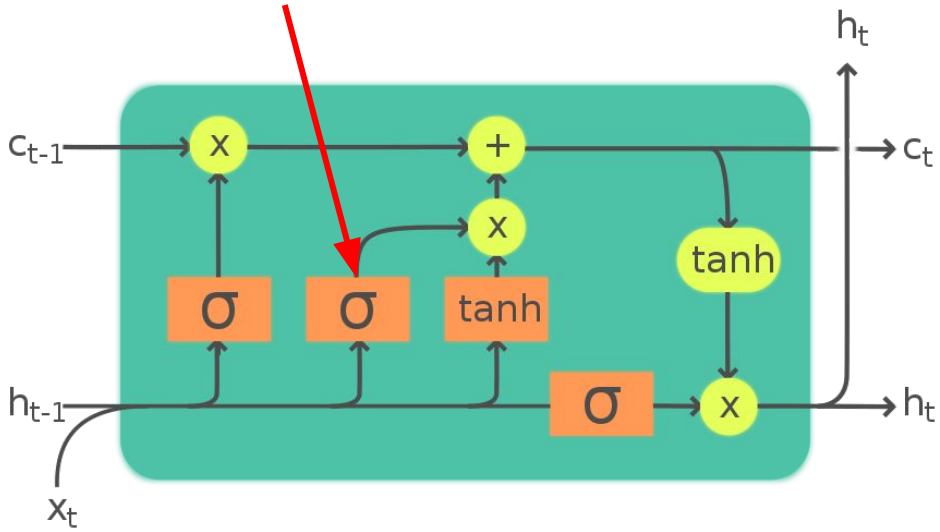
Забывающий вентиль

Здесь определяется % информации из c_{t-1}, которую можно забыть.



Входящий (обновляющий) вентиль

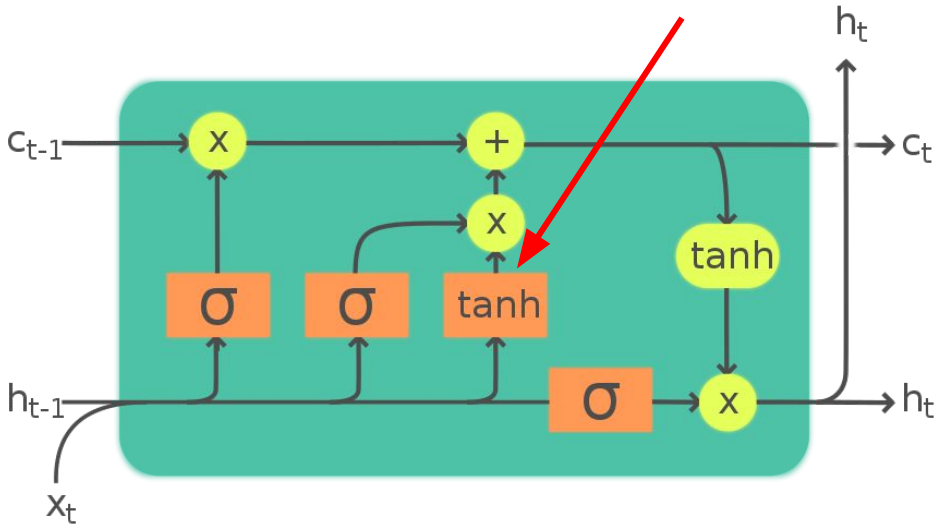
Здесь определяется % текущей информации (вход x_t), которая имеет долгосрочную релевантность и должна быть записана в c_t.



Нормировка данных

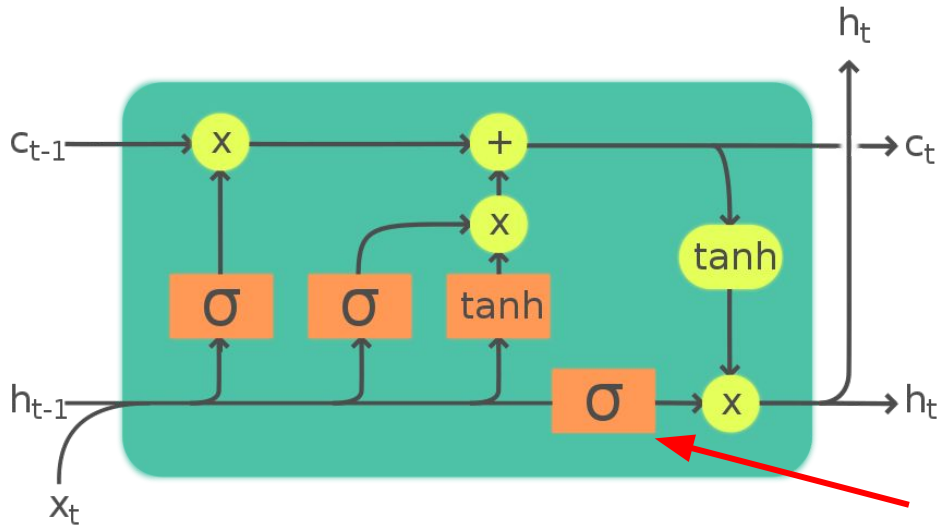
Здесь данные текущей ячейки (x_t, h_{t-1}) нормируются и выше записываются в долгосрочную память c_t.

Нормировка нужна для совпадения масштабов (единиц измерения) информации из разных слоев РНС.



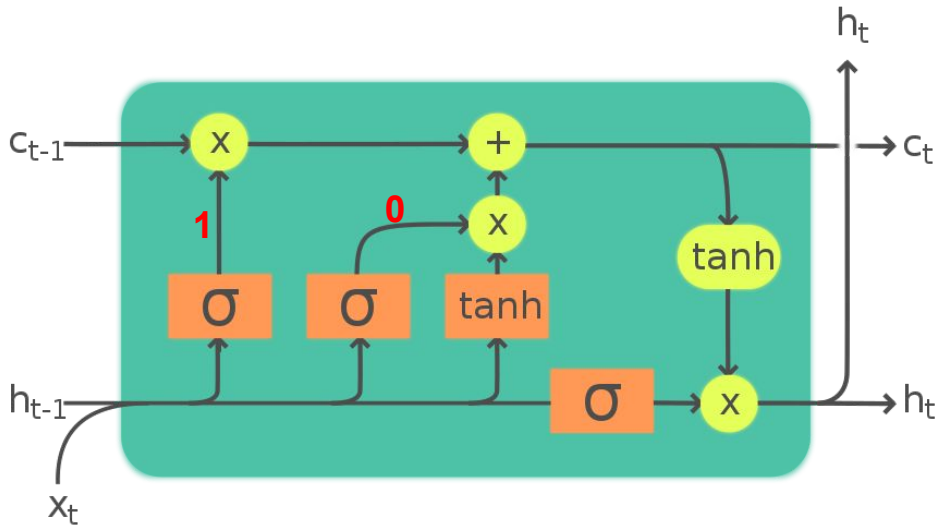
Выходной вентиль краткосрочной памяти

Здесь определяется степень релевантности краткосрочной памяти h_t — какой % краткосрочной памяти передать дальше.



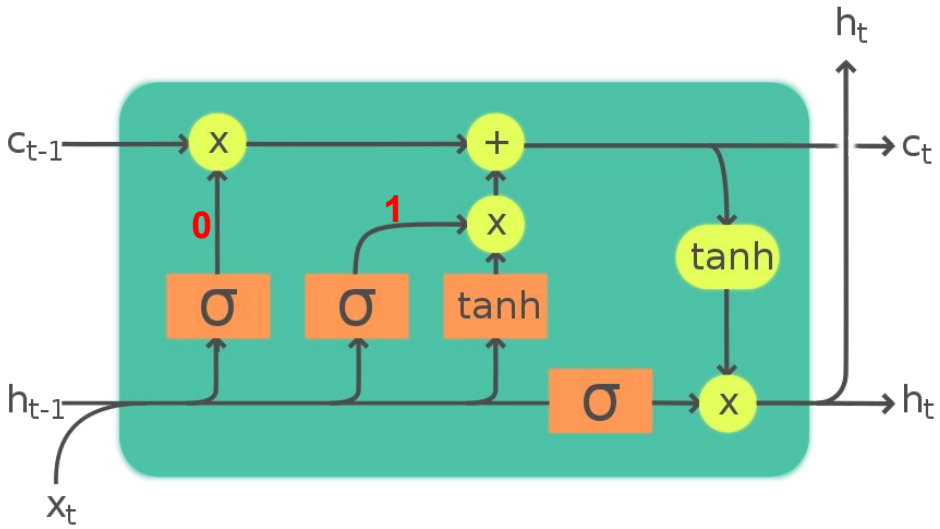
Порция общих слов

В процессе тренировки может случиться, что по указанным каналам придут значения 1 и 0. В этом случае долгосрочная память c_t вообще никак не изменяется в текущей ячейке LSTM.



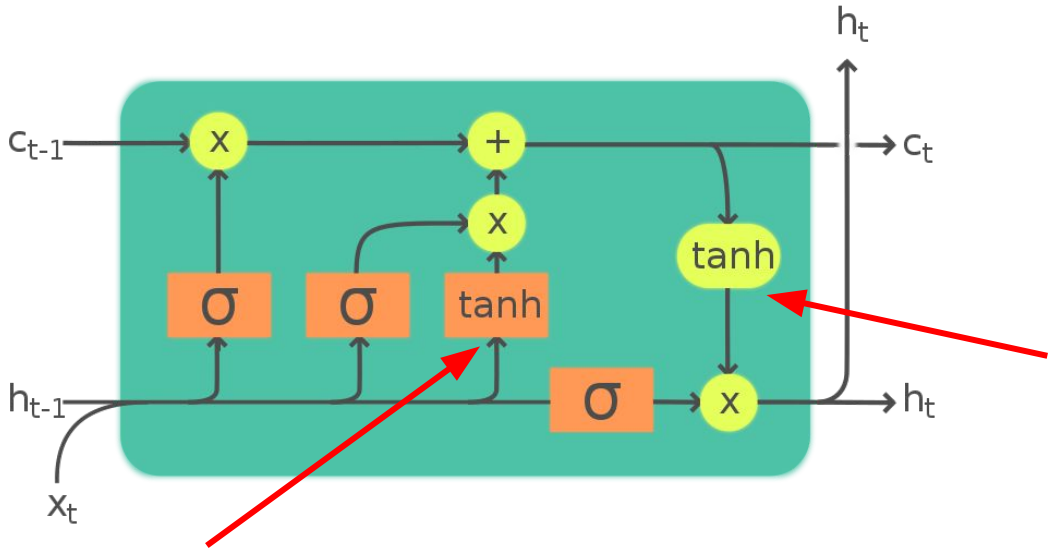
Порция общих слов

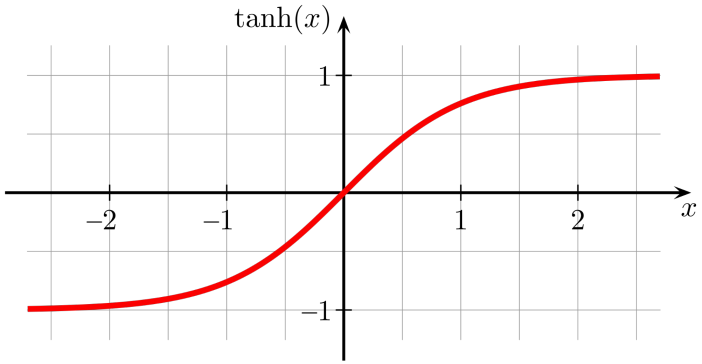
И наоборот: при таких значениях в каналах, вся информация с предыдущих слоёв потеряется.



Зачем тангенс? Да ещё и гиперболический!

Это функции активации.
Тангенс нормирует данные (это следует из его графика).
Сигмоида для нормировки не годится — будет затухание градиента в глубокой НС.

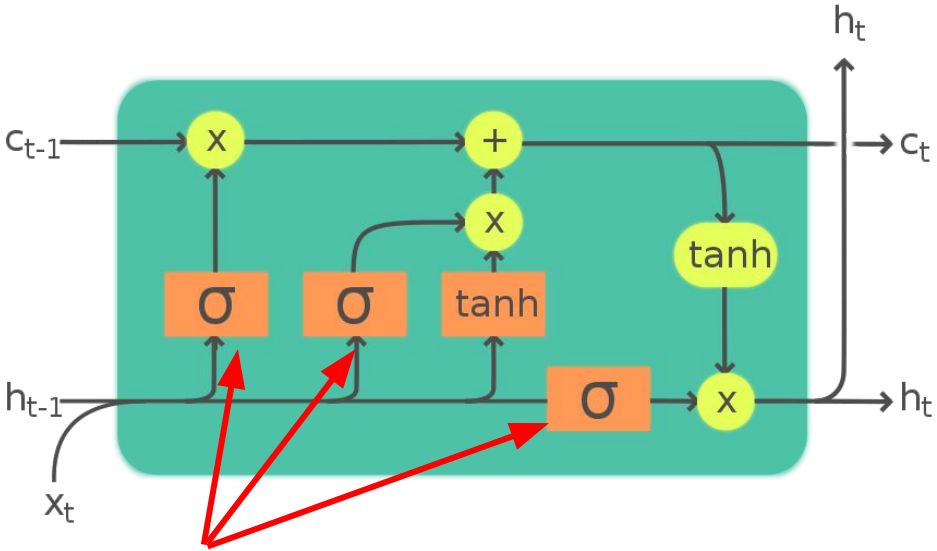


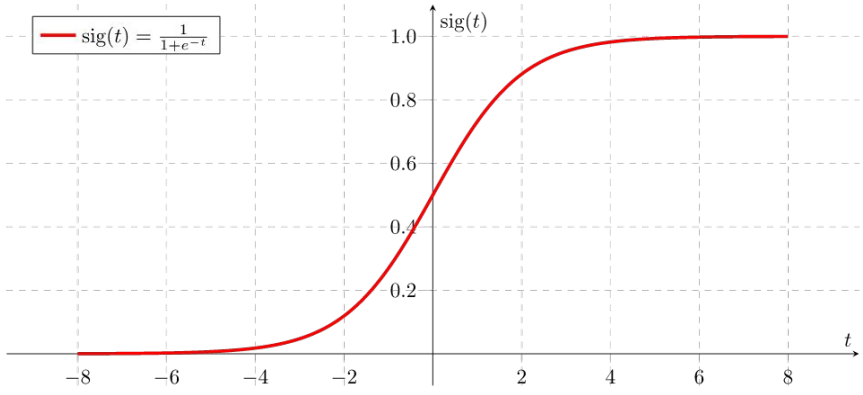


Почему же здесь сигмоиды?

Тут сигмоида нужна из-за её свойства **превращать вещественные числа в вероятности**.

Как было указано выше, эти узлы вычисляют разные % от входных данных.

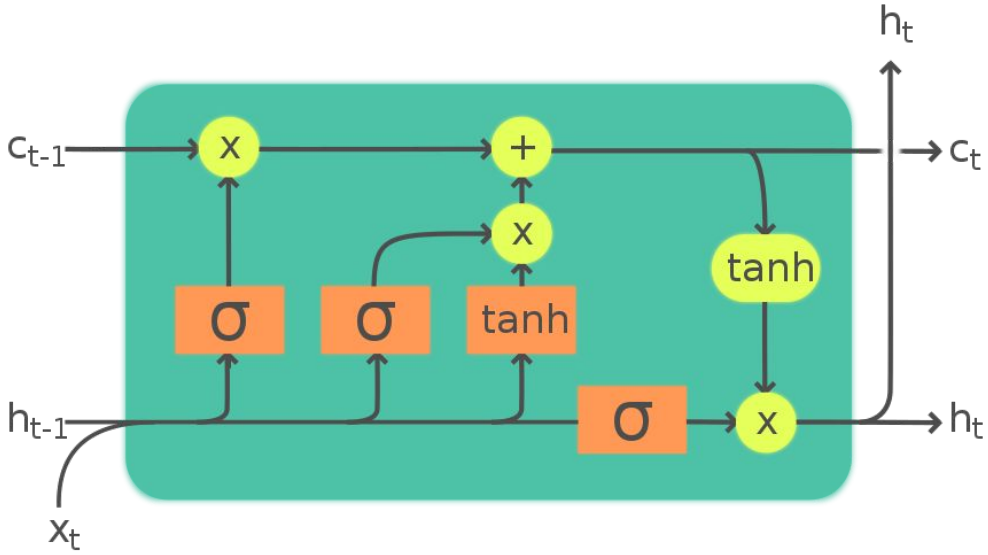




Веса LSTM-блока

У LSTM-блока много внутренних весов для тренировки.

Каждый оранжевый блок имеет свои веса для входящих в него величин h_{t-1}, x_t.

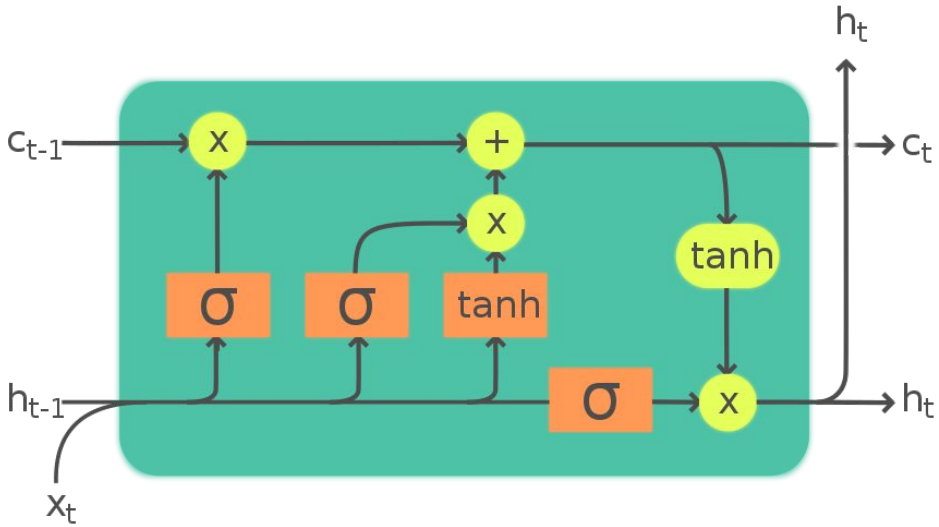


Вычисления LSTM-блока

Вы что-нибудь поняли?

Будем разбирать по частям.

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



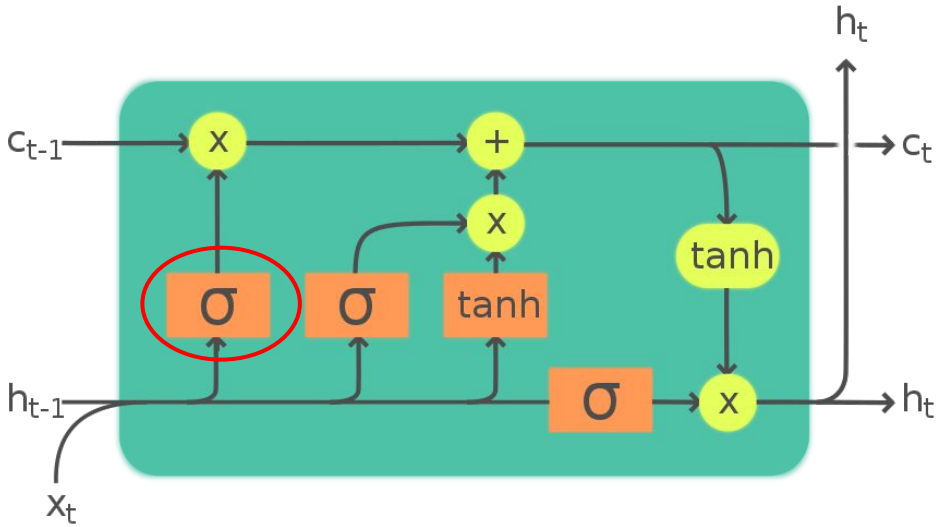
Вычисления LSTM-блока

Вычисление % забывания долгосрочной памяти.

σ_g — это сигмоида.

Тут **веса** для тренировки w_f, u_f, b_f.

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



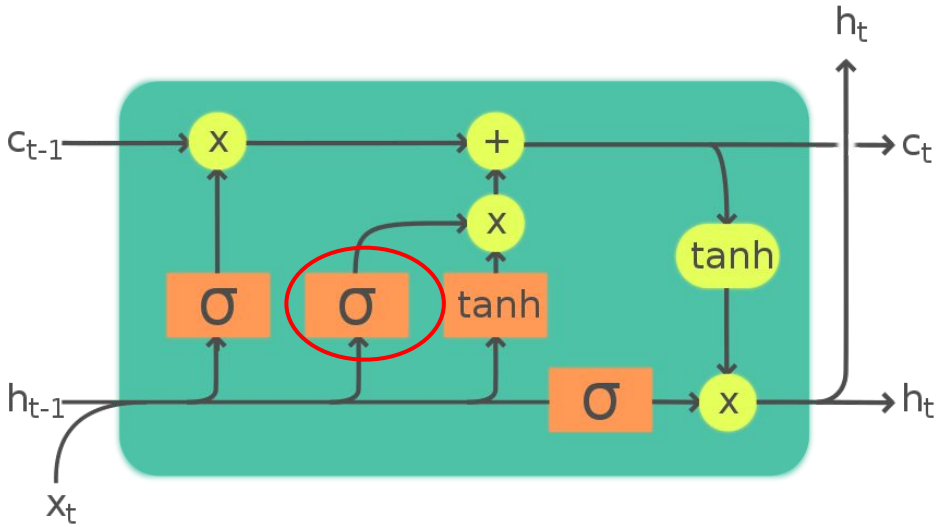
Вычисления LSTM-блока

Вычисление % текущей информации, которая имеет долгосрочную релевантность.

σ_g — это сигмоида.

Тут **веса** для тренировки w_i, u_i, b_i.

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



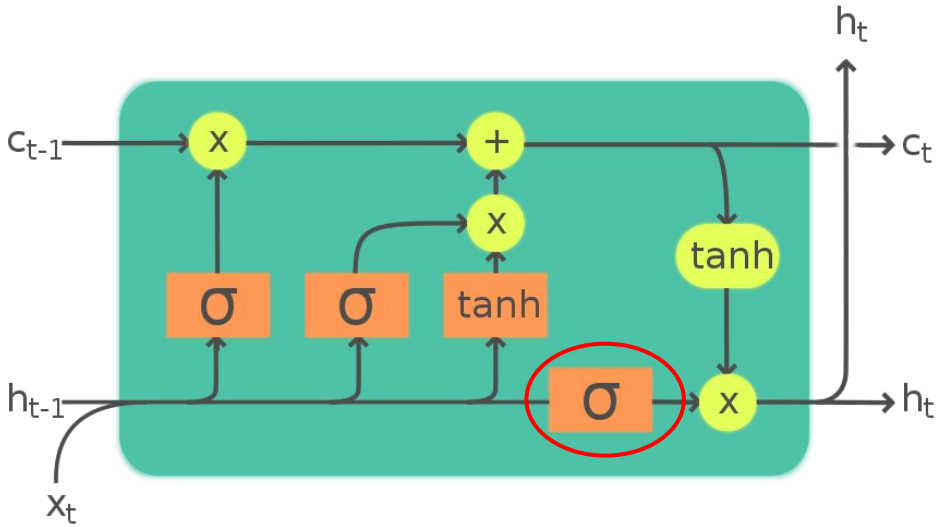
Вычисления LSTM-блока

Вычисление % краткосрочной памяти для передачи в следующих блок.

σ_g — это сигмоида.

Тут **веса** для тренировки w_o, u_o, b_o.

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



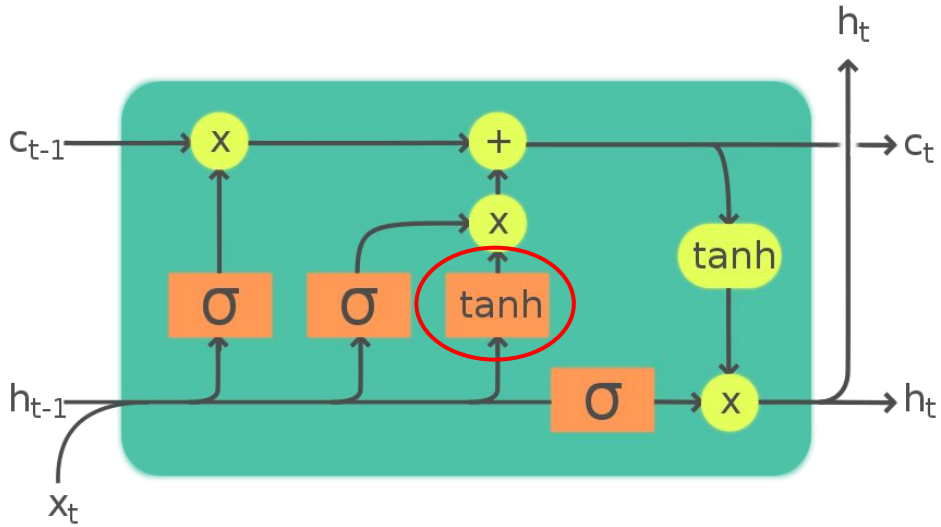
Вычисления LSTM-блока

Умная нормировка данных для записи в долгосрочную память.

σ_c — это гиперболический тангенс.

Тут **веса** для тренировки w_c, u_c, b_c.

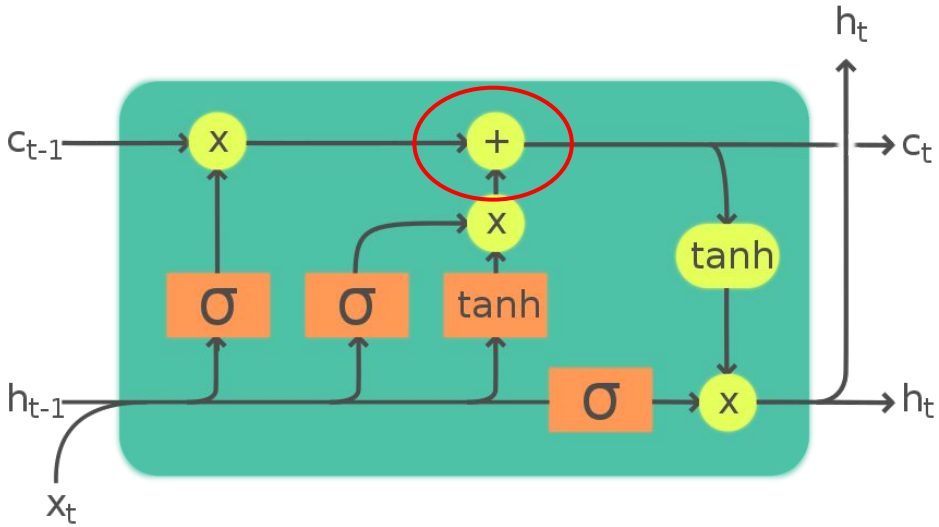
$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$



Вычисления LSTM-блока

Обновление долгосрочной памяти, применяем операции забывания и записи текущих данных в долгосрочную память.

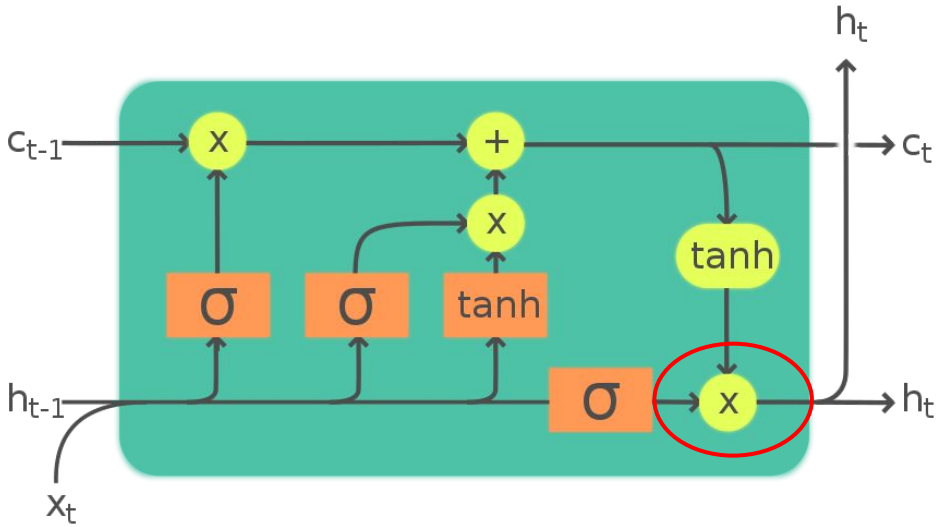
$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$



Вычисления LSTM-блока

Обновление краткосрочной памяти.

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

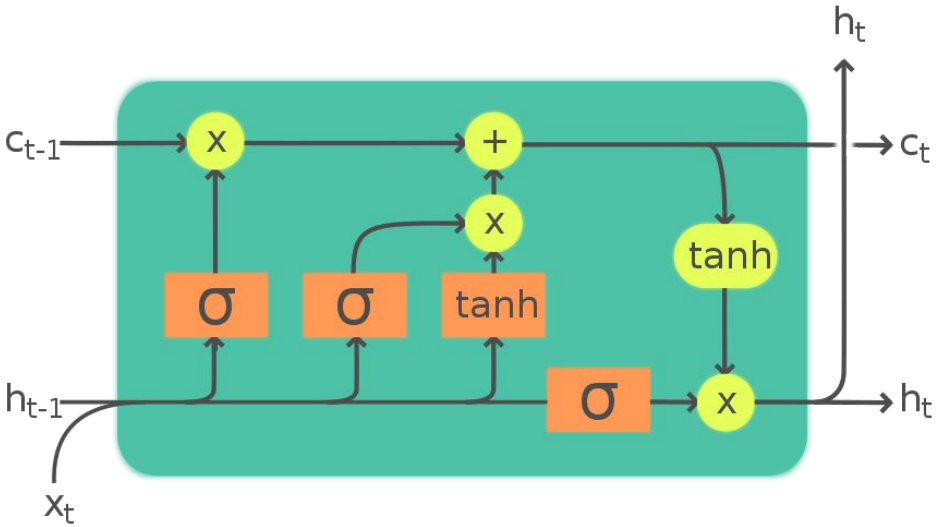


Вычисления LSTM-блока

Как видно, блок LSTM содержит 12 весов: w_f, u_f, b_f, w_i, u_i, b_i, w_o, u_o, b_o,w_c, u_c, b_c.

И это для одномерных x_t, h_t, c_t!

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

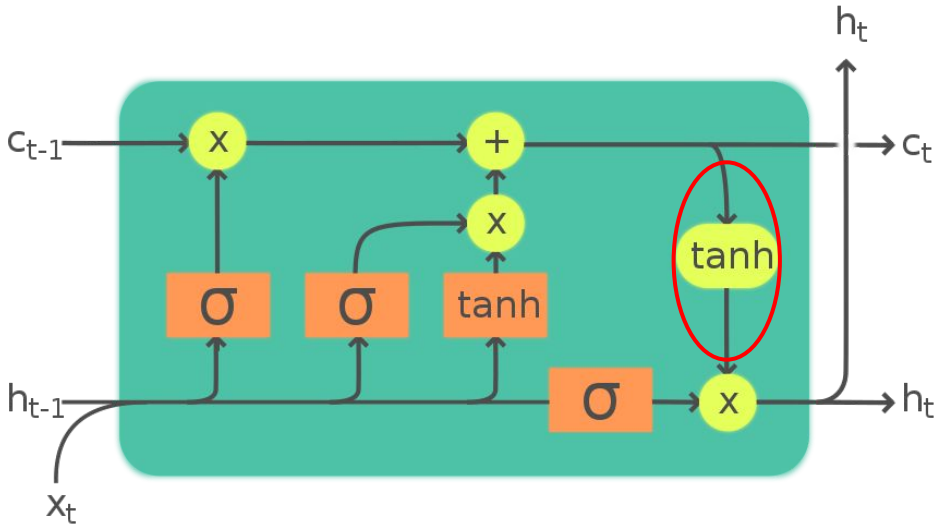


Остался вопрос: зачем эта перемычка?

Получается, что долгосрочная память влияет на краткосрочную? А разве у людей не так? Помните номер автобуса, на котором вчера приехали домой?

Но есть и чисто математическое объяснение…

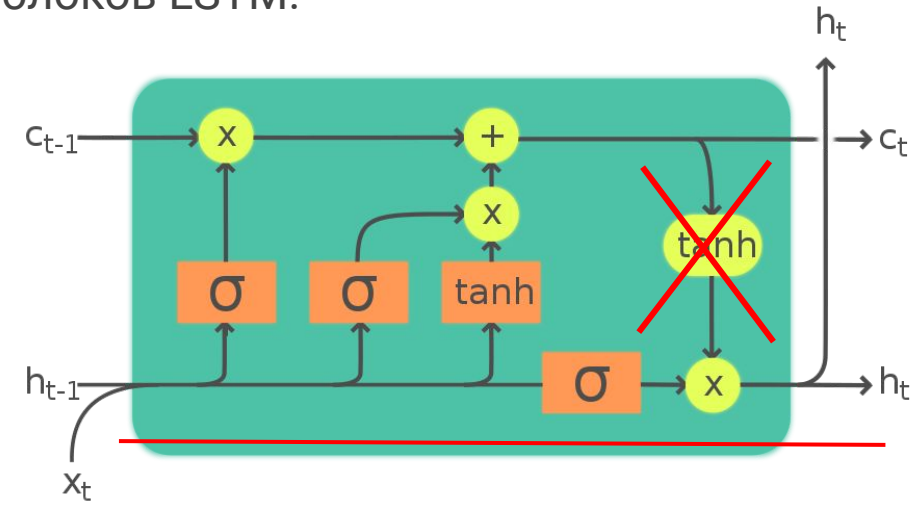
$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$



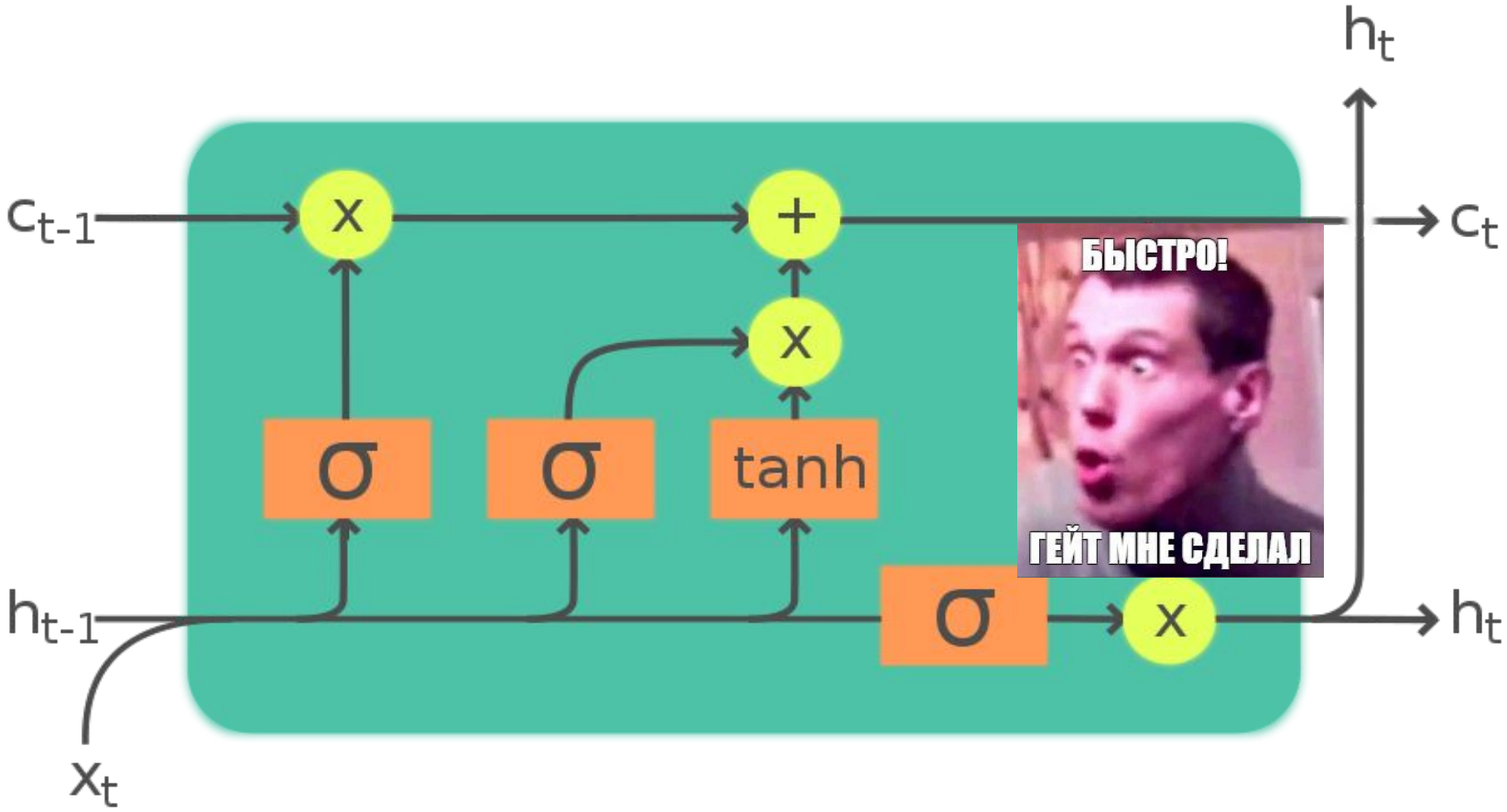
Остался вопрос: зачем эта перемычка?

Допустим, что перемычки нет. **Будет только хуже**. А почему?
Посмотрим на движение сигнала по «нижней магистрали». Там стоит функция активации «сигмоида». От неё не избавиться, поскольку она вычисляет % сохранения информации. После нескольких блоков LSTM значение h_t пройдет через композицию нескольких сигмоид. А это приведёт к затуханию градиента по h_t и невозможностью изменять веса блоков LSTM.

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



Так что сложная архитектура LSTM — не прихоть



Выводы

- Мы рассмотрели рекуррентную архитектуру нейронных сетей.
- Мы поняли, какую роль тут играет длина истории.
- Посмотрели на процесс тренировки таких сетей.
- Столкнулись с проблемами, но узнали про улучшенную архитектуру LSTM.