Конспект лекции

А.Н. Шевляков

Введение в нейронные сети

Рекуррентные нейронные сети, их применение для обработки последовательностей: текстов, аудио- и видеофайлов

# Обработка последовательностей

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.

Для обработки последовательностей используются рекуррентные нейронные сети (РНС).

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

В РНС всё не так: для них существенно важна история подачи объектов на вход РНС. Результат работы РНС для объекта А существенно зависит от объектов, которые были поданы ранее.

А это очень важно при обработке последовательностей.

Осталось понять, как это реализовать с помощью математики.

Давайте на вход НС будем подавать **два объекта**: объект А (который мы хотим обработать) и **выходное значение НС,** возникшее при обработке предыдущего объекта. То есть появляется архитектура подобного типа (Рис.1).

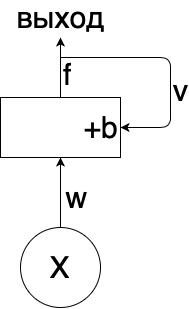


Рис. 1

Но есть проблема: как у такой НС вычислить функцию сети FNN(x)?

FNN(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?

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 x1,...,xt. В этом случае рекурсия остановится через t шагов.

Пусть t=3. Тогда функция сети для объекта x3 равна:

FNN(x3)=f(x3w+b+v)=f(x3w+b+vf(x2w+b+v))=f(x3w+b+vf(x2w+b+vf(x1w+b+0)))

Мы предполагаем, что вклад более ранних объектов в ответ для объекта х3 нулевой. Поэтому в нашей формуле и возникает 0.

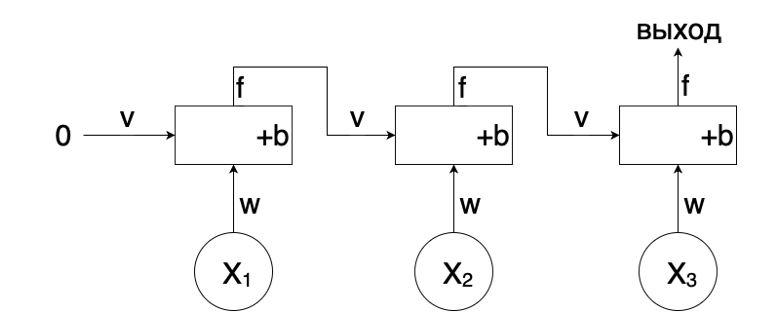
Переменные в этой формуле нужно интерпретировать так:

х3 - текущий объект,

х2 - предыдущий объект,

х1 - пред-предыдущий объект.

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



С функцией сети: FNN(x3)=f(x3w+b+vf(x2w+b+vf(x1w+b+0)))

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

FNN(x2)=f(x2w+b+vf(x1w+b+0))

FNN(x1)=f(x1w+b+0)

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

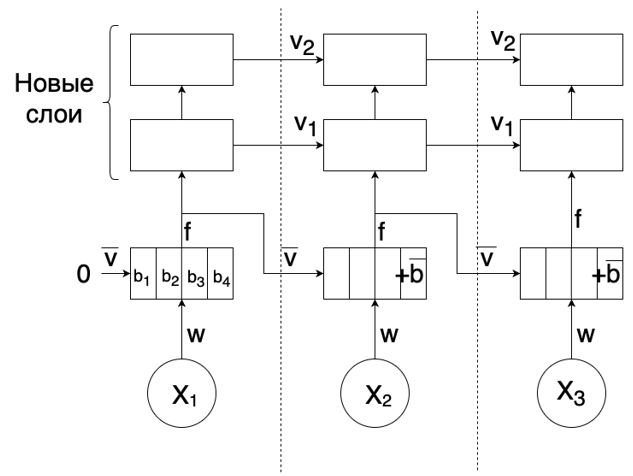
Входной объект xi всегда домножается на **один и тот же** параметр w; во всех нейронах прибавляется **одно и то же** смещение b; всюду **одна и та же** функция активации f, **одно и то же** число v.

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

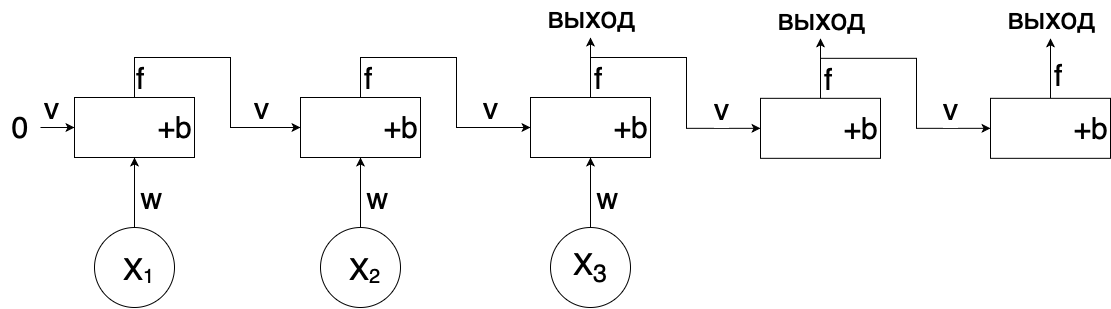
Рассмотренный выше простейший тип РНС был самым простым. Можно увеличивать число слоёв и длину истории.

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.

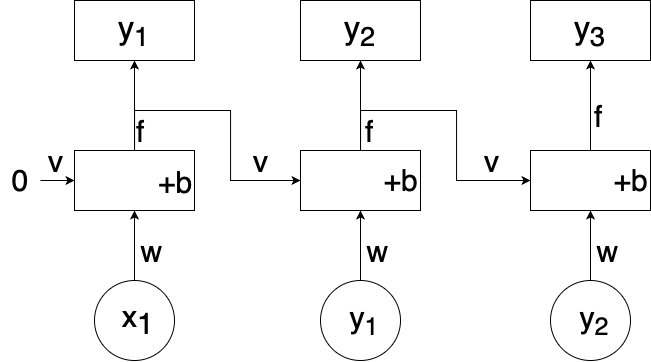


Можно сделать архитектуру, которая сначала «всасывает» в себя входные объекты, а потом генерирует последовательность выходов. Это используется, например, при автоматическом переводе текста (сначала загружаем в РНС весь текст, а затем получаем перевод).



# 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.

x1,y1,y2,y3 — слова нашего языка.

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».

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.

Минимизируем функцию потерь, находим оптимальные значения весов РНС. Как теперь генерировать текст? Придумываем какое-нибудь первое слово текста. Например: x1=«старуха».

Пропускаем его только эту часть РНС. Сеть выдает новое слово текста. Например: y1=«процентщица». Далее процедура повторяется: подаём на вход слово y1=«процентщица» и получаем новое слово.

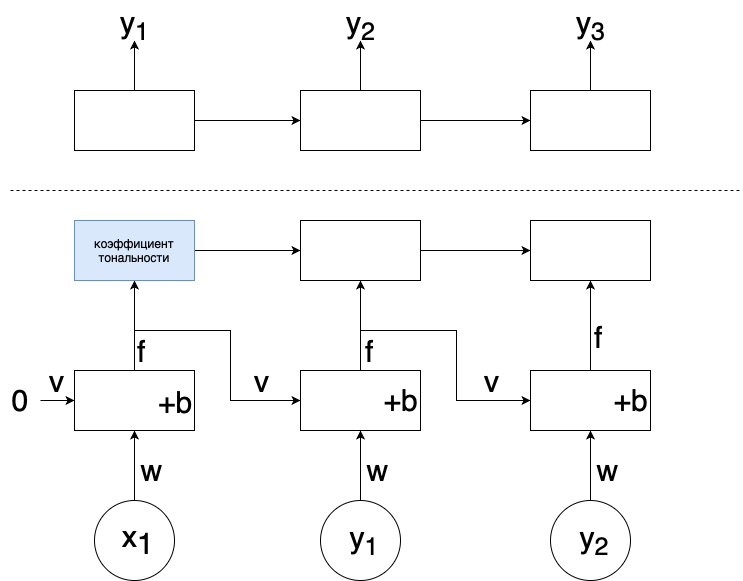
Ну и если вам повезёт, то получится текст «в духе Достоевского». Совпадение сюжетных линий, естественно, не гарантируется.

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.



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. 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.

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.

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

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…

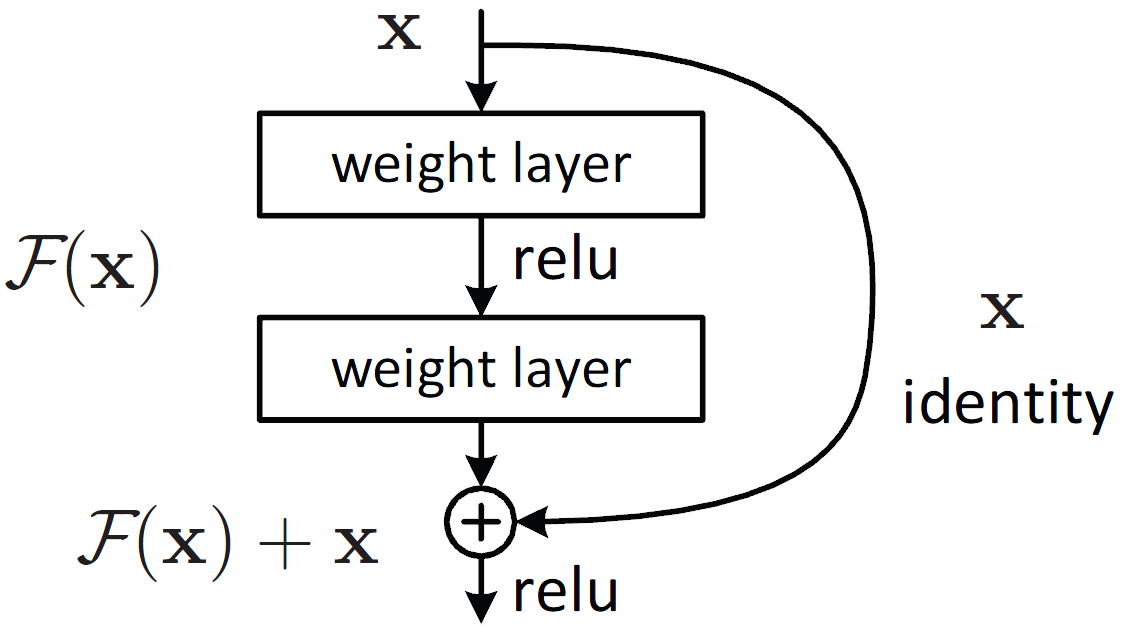
Человек дополнит стих на основе ритма, а ИИ должен обратиться к истории и **фактически выполнить вычитание.** Это единственный способ для ИИ закончить стих. Если доступа к далекой истории нет, то ИИ может основываться только на статистических (частотных) соображениях: «Что-то здесь мальчиков многовато. Для выправления процентного соотношения нужно добавить девочек. Поэтому закончим так… **Наташа**».

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.

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.

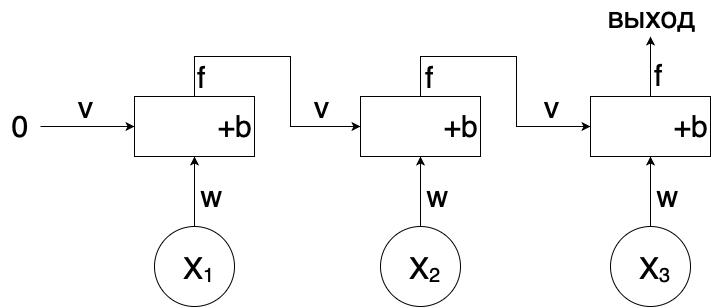
Конечно, есть методы борьбы с затуханием градиента. Например, идея из архитектуры СНС ResNet позволяет проталкивать информацию без искажений через сколь угодно большое количество слоев СНС.



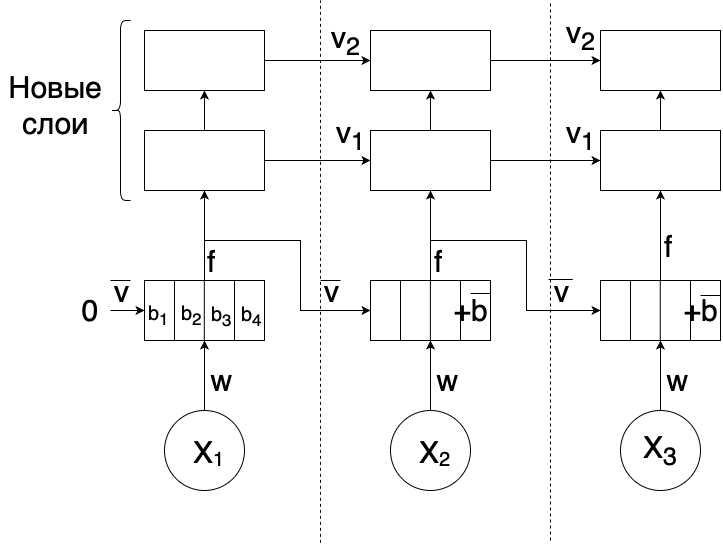
Но в случае с РНС возникает проблема. Дело в том, что у РНС много входов (количество входов равно длине истории).

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

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



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.



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);
* 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!

**Для простоты**: далее все РНС работают с **данными размерности 1х1**. То есть на вход РСН подаются одиночные числа, и между слоями РНС передаются тоже числа.

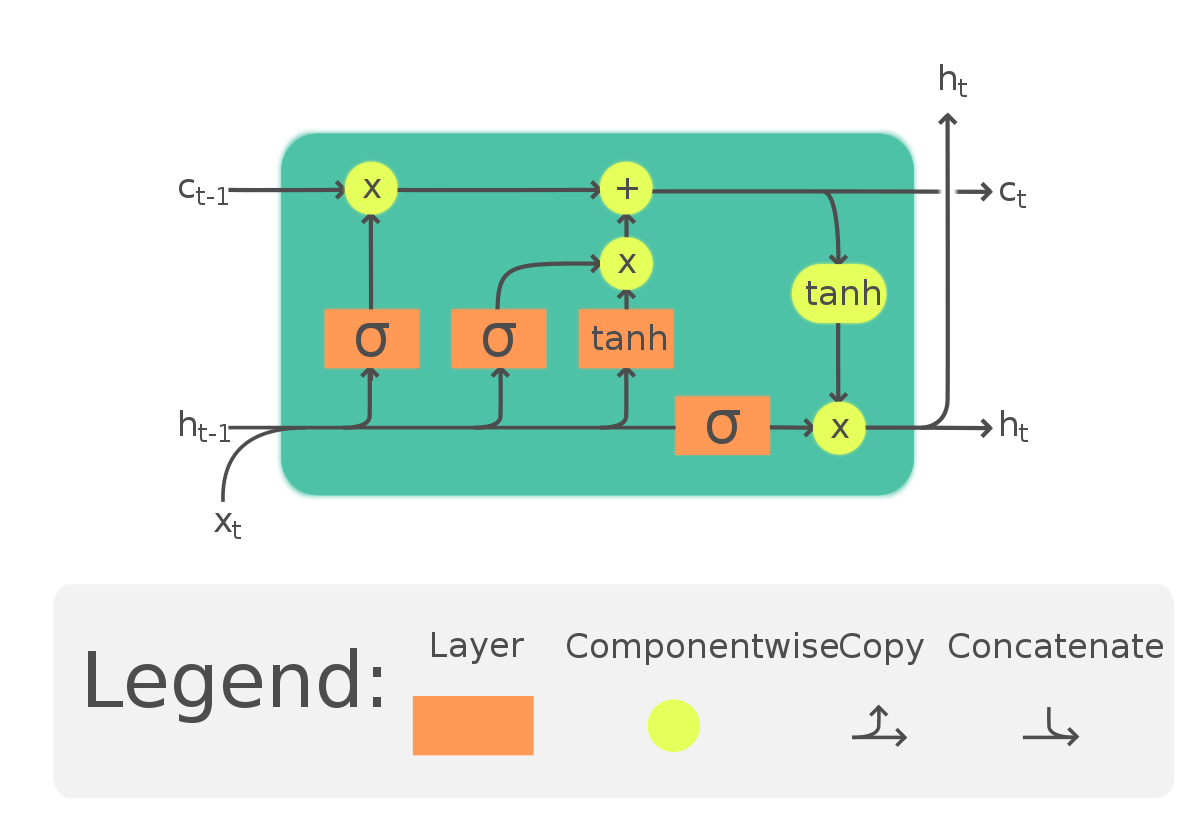
В стандартной архитектуре РНС с предыдущего слоя передается результат вычислений, умноженный на вес v.

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

Перед первым слоем РНС логично предположить, что h0=c0=0.

Их смысл: ht — краткосрочная память LSTM (это полный аналог веса **v**),

ct — долгосрочная память LSTM

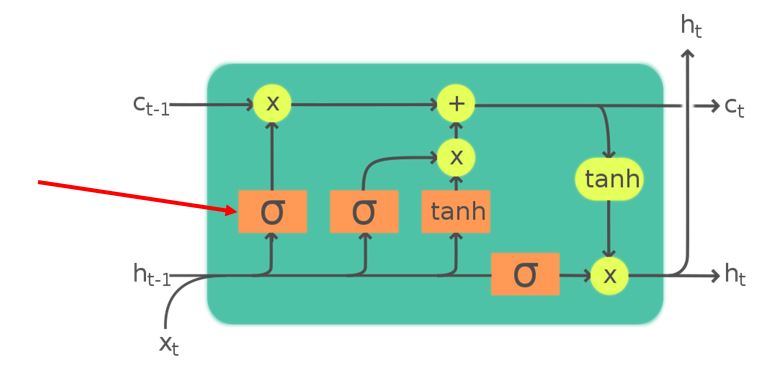


Мы видим, что всю РНС пронизывает «магистральная связь», по которой значения ct из далёких ячеек могут быть без изменения переданы дальше.

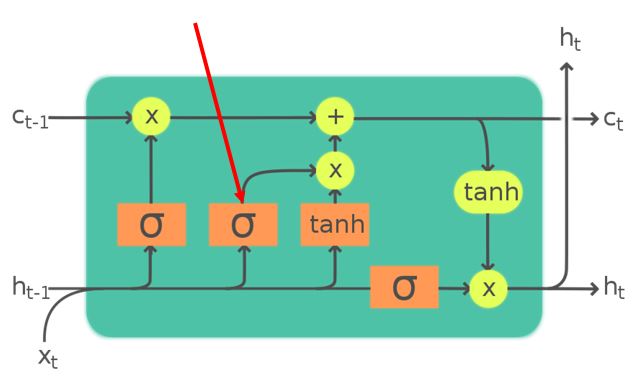
Это будет при условии, что LSTM-блок не захочет изменять ct. Степень изменения значения ct внутри очередного блока LSTM определяется в процессе тренировки РНС.

Из чего состоит LSTM-блок? Сначала рассмотрим, основные узлы (вентили, гейты) LSTM-блока.

**Забывающий вентиль**. Здесь определяется % информации из сt-1, которую можно забыть.

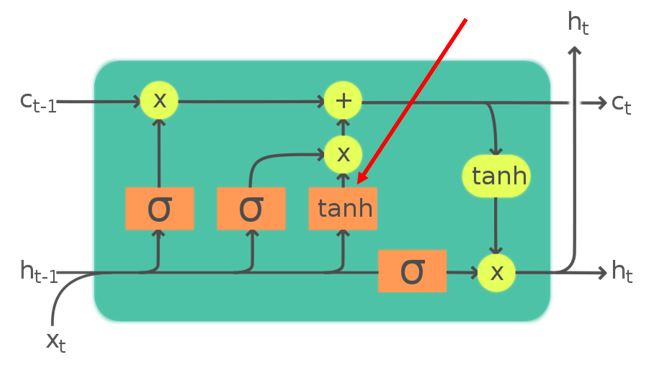


**Входящий (обновляющий) вентиль.** Здесь определяется % текущей информации (вход xt), которая имеет долгосрочную релевантность и должна быть записана в ct.

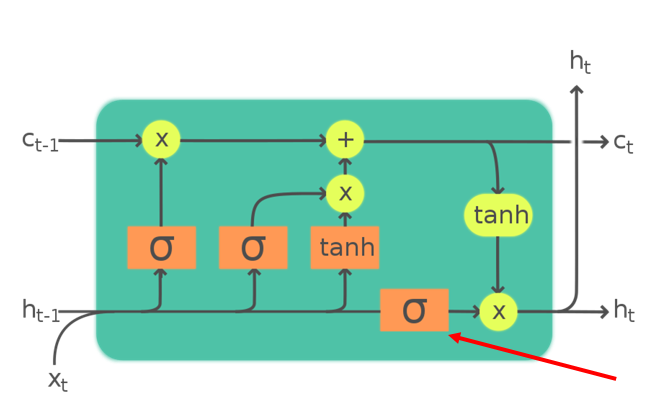


**Нормировка данных.** Здесь данные текущей ячейки (xt, ht-1) нормируются и выше записываются в долгосрочную память ct.

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



**Выходной вентиль краткосрочной памяти.** Здесь определяется степень релевантности краткосрочной памяти ht — какой % краткосрочной памяти передать дальше.



В процессе тренировки может случиться, что по указанным каналам придут значения 1 и 0. В этом случае долгосрочная память ct вообще никак не изменяется в текущей ячейке LSTM. И наоборот: при таких значениях в каналах, вся информация с предыдущих слоёв потеряется.

Тангенс нормирует данные (это следует из его графика). Сигмоида для нормировки не годится — будет затухание градиента в глубокой НС. Тут сигмоида нужна из-за её свойства **превращать вещественные числа в вероятности**.

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

У LSTM-блока много внутренних весов для тренировки. Каждый оранжевый блок имеет свои веса для входящих в него величин ht-1, xt.