ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
"МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. ЛОМОНОСОВА"

ФИЗИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА МАТЕМАТИКИ

ПРЕДСКАЗАНИЕ ВРЕМЕННЫХ РЯДОВ ПРИ ПОМОЩИ НЕЙРОННЫХ СЕТЕЙ

Выполнил студент
211 группы
Попов Иван Васильевич

_____
подпись студента

Научный руководитель:
Доктор физико-математических наук
Голубцов Петр Викторович

_____
подпись научного руководителя

Москва
2022 год

Содержание

Введение

Что такое искусственный интеллект? Это не робот, который имеет самосознание и может захватить мир или просто действовать как живое существо. На данный момент искусственный интеллект – это прикладные программы, которые умеют решать конкретные задачи: распознавать лица, речь, водить машину, определять заболевания по симптомам и так далее. Важно понимать, что нейронные сети – не аналог искусственного интеллекта, а лишь один из способов его реализации.

На протяжении последних 100 лет многие умы пытались создать искусственный интеллект, однако для этого требовались огромные вычислительные мощности. С ростом количества информации и данных, обрабатываемых компьютером, неудивительно, что сейчас так популярны нейронные сети и глубокое обучение. Их прогресс зашел так далеко, что в некоторых областях нейросети справляются с задачей лучше человека. Например, нейронные сети, обученные на наборе данных ImageNet, выдают ошибку в долях процента при распознавании образов на картинке, тогда как человек в среднем ошибается в 5% случаях на тех же изображениях.

Чем же полезны нейронные сети при обработке временных рядов? Приведем несколько примеров. Нейросети могут строить предсказания на основе обрабатываемых данных, будь то оборот компании, спрос товаров, трафик на сайте, курсы акций или валют. В некотором смысле мы можем заглянуть в будущее. Во-вторых, сейчас широко изучается применение нейросетей в биометрике. С их помощью можно выявить отклонения в здоровье человека и даже читать мысли, исследуя электроэнцефалограмму.

Цель работы

Изучение временных рядов и их прогнозирование при помощи рекуррентных нейронных сетей. Исследование LSTM и GRU моделей.

Глава 1. Нейронные сети

1.1 Математическая модель нейрона.

Для описания математической модели нейрона нужно вспомнить несколько фактов из школьного курса биологии, а именно – строение нашей нервной системы.

Нервная система человека – это самый большой и эффективный обработчик информации в человечском теле. Она состоит из нескольких компонент:

1. мозг, который отвечает за обработку информации и принятие решений;
2. органы чувств. При их помощи в мозг поступает информация по нервным волокнам;
3. мышцы. К ним передаются сигналы от мозга.

Все эти компоненты состоят из примерно одинаковых клеток, которые называются нейронами (рис. 1). У нейрона есть: ядро (тело нейрона) – в нем накапливается электрический заряд, дендриты – по ним передается сигнал от других нейронов, аксон – по нему передается сигнал к другим нейронам. Место соединения дендрита с аксоном называется синопсом. Он может быть сильным (сигнал полностью переходит к нейрону), а может быть слабым (заряд практически не проходит). В зависимости от обстоятельств, синопс может становиться слабее, либо сильнее. Именно с настройкой синопса

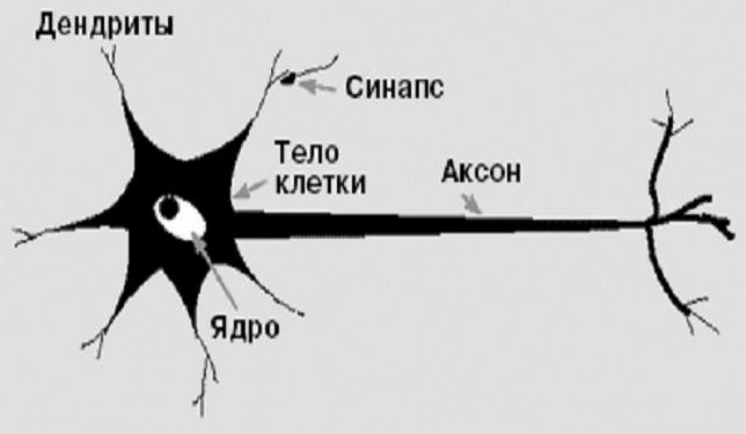

Рис. 1. Биологическая структура нейрона.

и связана биологическая тренировка нейронной сети и, соответственно, нервной системы человека.

Опишем математическую модель, основанную на этих идеях. На рисунке 2 приведена модель одного нейрона. Тело нейрона заменяется на

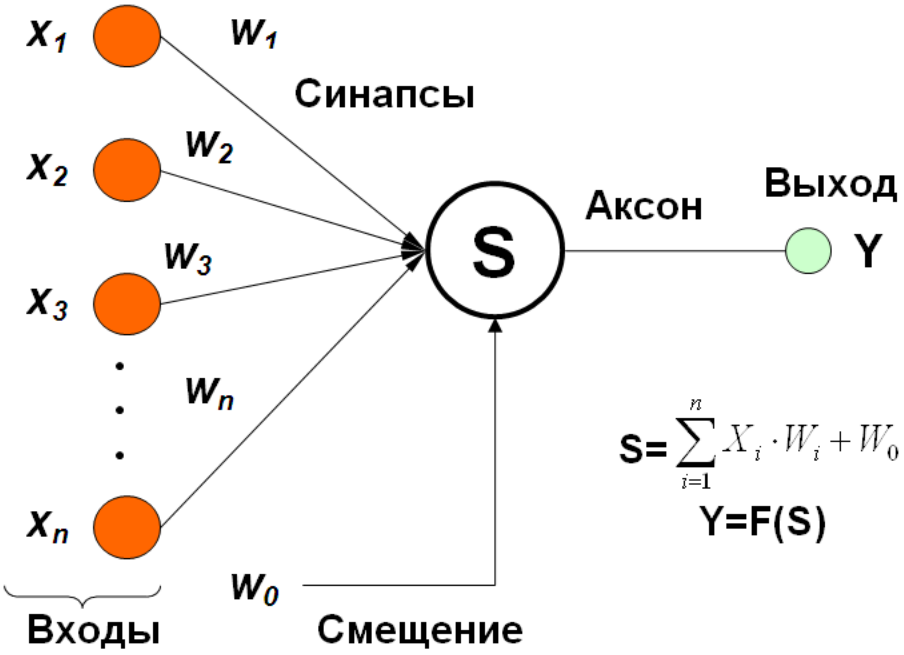

Рис. 2. Математическая модель нейрона.

сумматор (на рисунке имеет обозначение S). Также у нейрона есть еще и дендриты, по которым сигнал поступает в тело от других нейронов. Входы в сумматор будут заменять нам дендриты.

Кроме того, у нейрона есть аксон, по которому сигнал переходит к другим нейронам. В данной модели эту функцию выполняет выход. Биологический нейрон обрабатывает сигналы, которые в него приходят, а именно, он накапливает заряд до тех пор, пока он не достигнет какого-то значения, и только после этого заряд уходит по аксону к другим нейронам. Это реализуется посредством функции активации F(S). Перед тем, как отправить сигнал к другим нейроном, мы будем пропускать значение из сумматора через функцию.

Еще одним важным компонентом модели являются синапсы. Они отражают силу, с которой наш нейрон связан с другими. Эту силу мы можем

моделировать при помощи синаптических весов. Рассмотрим, как работает модель с несколькими входами.

Пусть у нас есть три входа: x_0, x_1, x_2. Далее вычисляется взвешенная сумма и передается в сумматор S:

$$S = x_0 * w_0 + x_1 * w_1 + x_2 * w_2 + w_3,$$

где w_0, w_1, w_2 – настраиваемые веса;

$w_3 -$ смещение.

После этого значение из сумматора пропускается через функцию F, которую принято называть функцией активации, и передается следующему нейрону в качестве входа y:

$$y = F(S) = F(x_0 * w_0 + x_1 * w_1 + x_2 * w_2 + w_3).$$

Отметим, что веса и смещение – настраиваемые параметры. В процессе обучения они будут корректироваться.

1.2 Обучение нейронных сетей.

Усложним нашу модель. На картинке ниже представлена полносвязная нейронная сеть. Это сеть, в которой каждый нейрон связан со всеми нейронами, находящимися в соседних слоях. Мы будем рассматривать

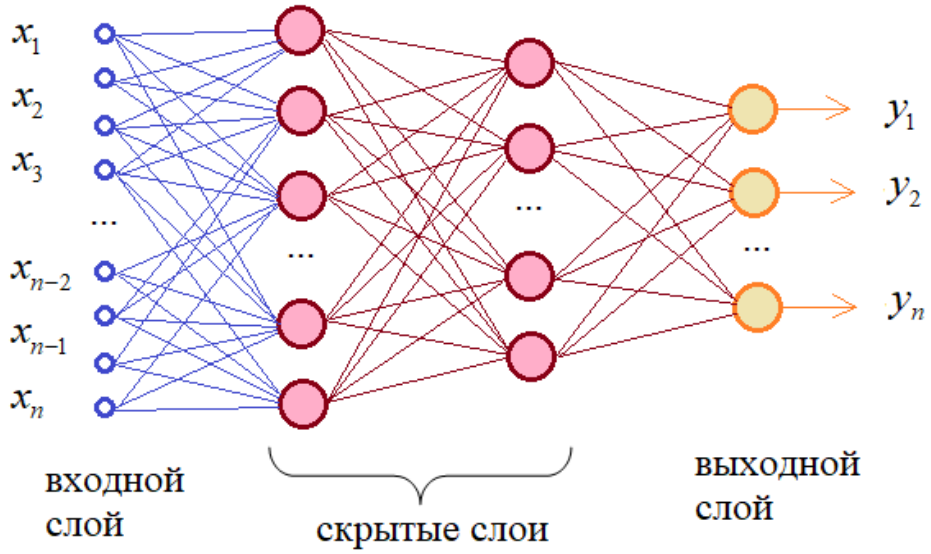

Рис. 3. Полносвязная нейронная сеть.

последовательные сети прямого распространения, т. е. сети, у которых все связи направлены только от входа к выходу.

Перед построением более сложных архитектур нейронных сетей сначала разберемся с настройкой весов или обучением нейронной сети в таком простом случае.

Чаще всего нейронные сети обучаются при помощи метода обучения с учителем. Это значит, что мы предоставляем нейронной сети примеры с известными правильными ответами. Такого типа данные называются обучающей выборкой и состоят из пар: входной объект и соответствующий ему желаемый правильный ответ. Однако задача нейронной сети – это не запомнить обучающую выборку (**эффект переобучения**), а научиться искать закономерности в данных, обобщать и делать хорошие предсказания на новых данных, которые нейросеть не видела во время обучения.

Итак, цель процесса обучения – найти такие параметры нейронной сети, при которых нейросеть ошибается меньше всего. Введем функцию потерь, которая будем показывать, как сильно ошибается нейронная сеть. Для прогнозирования временных рядов в качестве функции потерь подойдет среднеквадратическая ошибка (**MSE**):

$$E = \frac{1}{N}\sum_{i}^{N}(y_i - z_i)^2$$

где y – истинные значения; z – предсказания сети.

Таким образом, задачей обучения нейронной сети является минимизация функции потерь относительно весов сети. Веса в таком случае должны удовлетворять равенству:

$$W_{opt} = \underbrace{argmin}_{W} E(W)$$

Искать минимум многомерной функции – сложная процедура, поэтому пользуются алгоритмом **градиентный спуск.** Разберем идею данного алгоритма.

Предположим, что мы находимся в точке W_0 и график функции ошибки от одного измерения весов выглядит так:

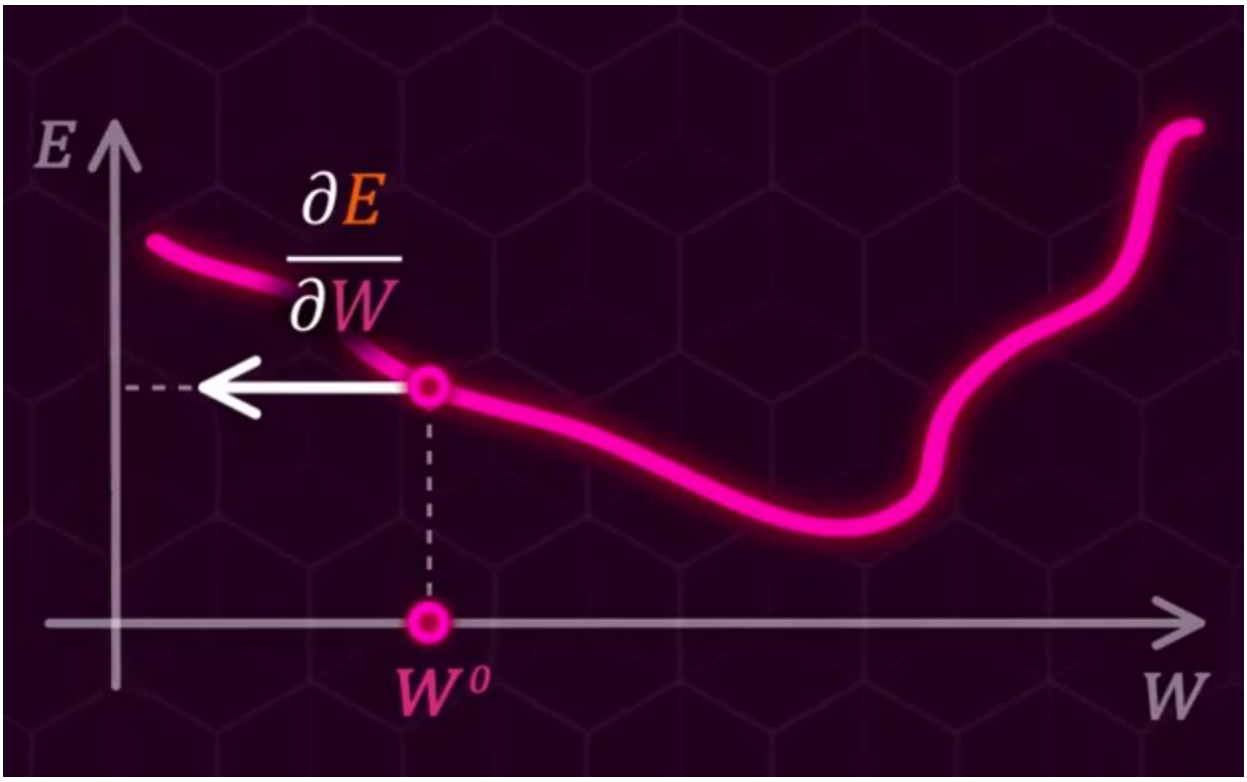

Рис. 4. Зависимость функции ошибки *E* от настраиваемого параметра *W*.

Тогда вектор градиента ошибки направлен в сторону локального возрастания функции ошибки. Значит, если мы будем изменять веса в противоположном направлении, то мы будем перемещаться в сторону локального минимума функции ошибки. Проделав несколько раз такую корректировку весов, мы окажемся в нужной точке. В виде формулы одна итерация градиентного спуска записывается так:

$$W_{t+1} = W_t - alpha * \frac{\partial E}{\partial W}(W_t)$$

где *alpha* – скорость обучения.

Распишем итоговый алгоритм обучения нейросети:

1. Инициализация начальных весовых коэффициентов W_0;
2. Берем из обучающей выборки группу обучающих образцов;

3. Подаем наши данные в сеть, то есть делаем прямое распространение, и получаем предсказание для текущих параметров;
4. Вычисление ошибки и градиента;
5. Делаем один шаг градиентного спуска вдоль антиградиента;
6. Повторяем пункт 2 с новыми параметрами сети. Останавливаемся, когда достигаем нужной ошибки.

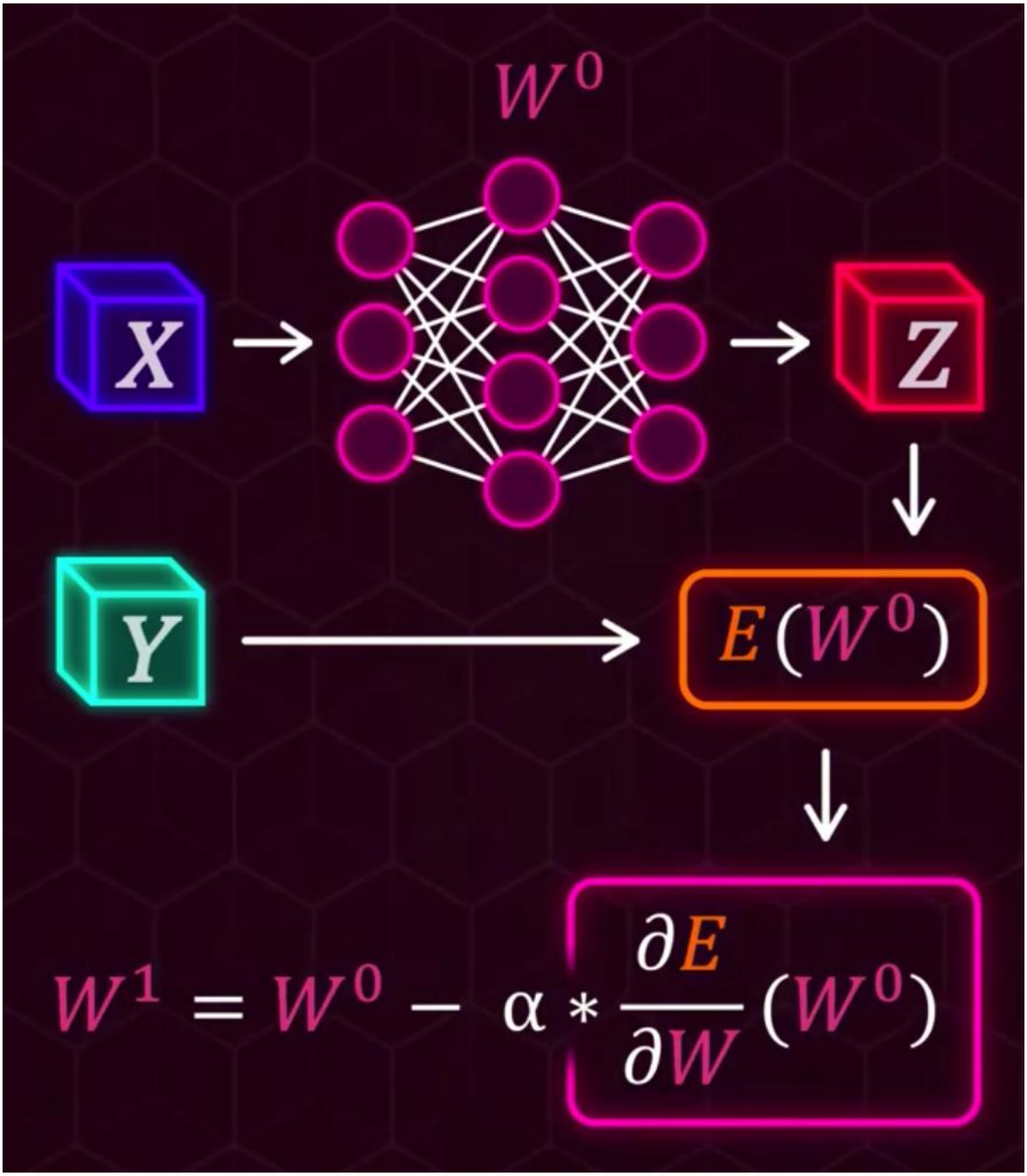

Рис. 5. Одна эпоха обучения нейронной сети.

Подробнее о том, как вычислять градиент написано в приложении 1.

1.3 Рекуррентные нейронные сети.

Главным недостатком сетей прямого распространения является сложность анализа временных последовательностей, например, звукового сигнала, изменение положения объекта во время движения и так далее. Здесь требуется знание предыдущих элементов последовательности. Для анализа таких данных были разработаны рекуррентные нейронные сети.

Рекуррентная нейронная сеть – это сеть с памятью, хранящая информацию о том, что в ней происходило в прошлые итерации обучения. Достигается это благодаря тому, что каждый нейрон в такой сети имеет связь с самим собой.

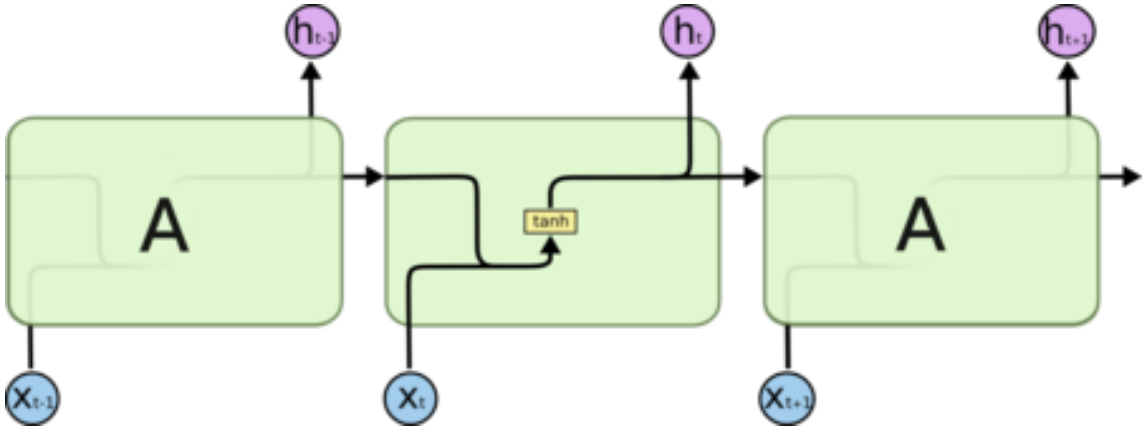

Рис. 6. Рекуррентная нейронная сеть.

Сначала сеть обрабатывает входные значения, пропускает взвешенные суммы через функцию активации, запоминает выходные значения и на следующей итерации подает их на вход вместе с новым вектором признаков объекта. Можно записать, чему равен вектор выходных значений нейронов скрытого слоя в момент времени t:

$$h_t = f(h_{t-1}, x_t)$$

Несложно догадаться, чему равен начальный вектор выходных значений $h_0 = [0, 0, 0, \dots, 0_N]^T$.

Рассмотрим возможные архитектуры таких сетей. На рисунке 7

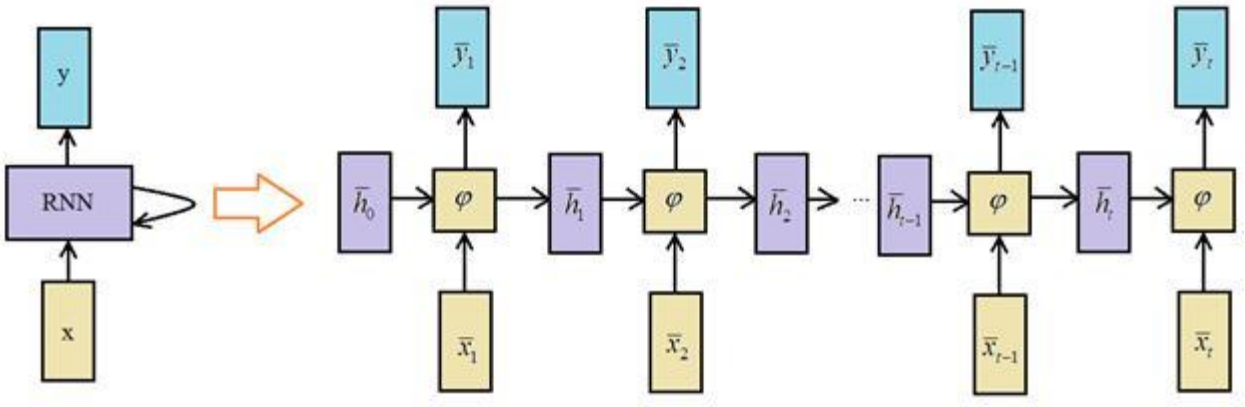

Рис. 7. Many to Many.

множеству входных векторов соответствует множество выходных, поэтому такая архитектура называется **Many to Many.** Используется для перевода текстов. По аналогии можно построить еще две архитектуры: **Many to One** и **One to Many.** Они приведены на рисунке 8.

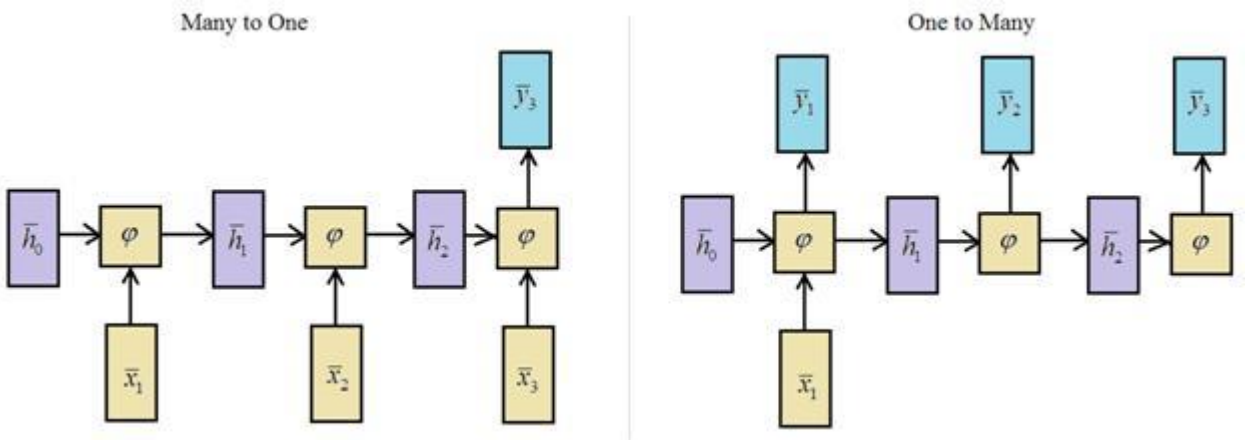

Рис. 8. Many to One и One to Many.

Many to One – множеству входных значений соответствует одно выходное. При их помощи можно анализировать эмоциональную окраску текста. One to Many – один вход и множество выходных значений. Пример использования – генерация описания изображений.

Рекуррентные нейронные сети обучаются так же, как и обычные полносвязные – при помощи метода обратного распространения ошибки. Для предсказания временных рядов будут использоваться архитектуры **Many to One.**

Глава 2. Применение нейронных сетей для предсказания временного ряда

2.1 Что такое временной ряд?

Временной ряд — это упорядоченная последовательность значений какого-либо показателя за несколько периодов времени. Основная характеристика, которая отличает временной ряд от простой выборки данных, — указанное время измерения или номер изменения по порядку.

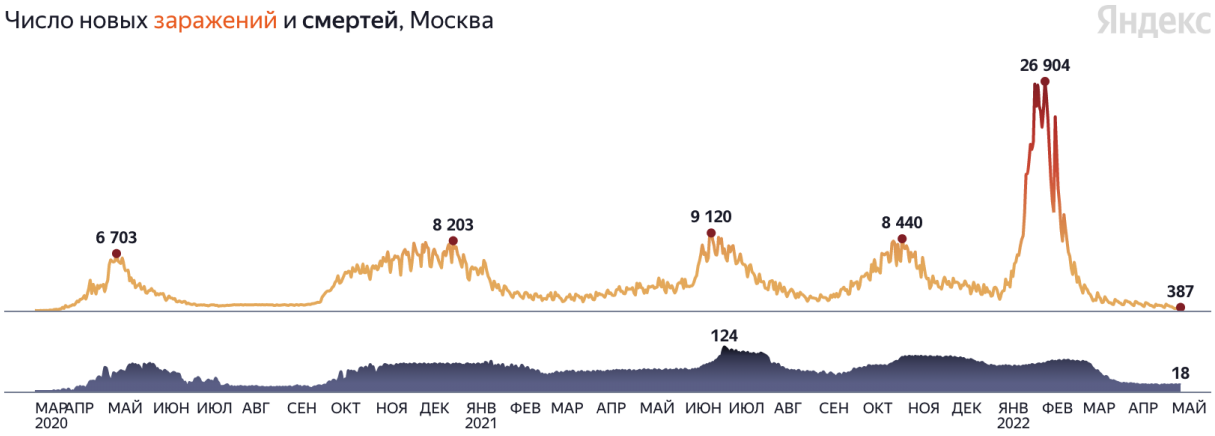

Рис. 9. Пример многомерного временного ряда.

Временные ряды бывают многомерными или одномерными. У многомерных рядов за каждый момент времени мы наблюдаем несколько показателей. На рисунке 9 собрана статистика заражений и смертей от коронавируса за каждый день. В данной работе будут анализироваться одномерные временные ряды.

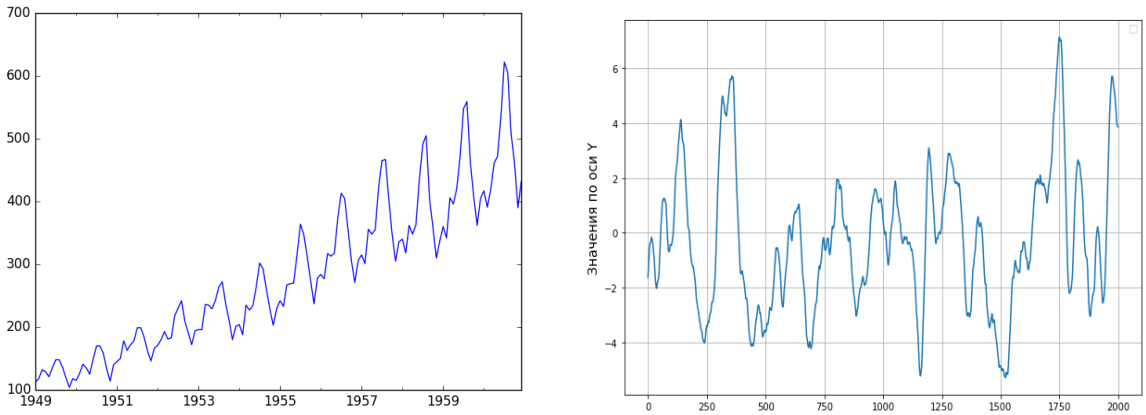

Рис. 10. Пример нестационарного(слева) и стационарного(справа) ряда.

Наложим еще одно требование на наши ряды. Они должны быть стационарными, т. е. не иметь тенденции или сезонных эффектов (среднее значение и дисперсия постоянны). Когда временной ряд является стационарным, его легче моделировать.

Разберемся, какие модели рекуррентных нейронных сетей помогут предсказывать временные последовательности.

2.2 Архитектура LSTM.

Основным недостатком простейших рекуррентных сетей является быстрое забывание прошлого контекста. Такая архитектура все смешивает в одну кучу и то, что встречалось вначале быстро теряется. Поэтому в 1997 году была предложена архитектура **LSTM** (Long short-term memory). Рекуррентный блок LSTM выглядит следующим образом:

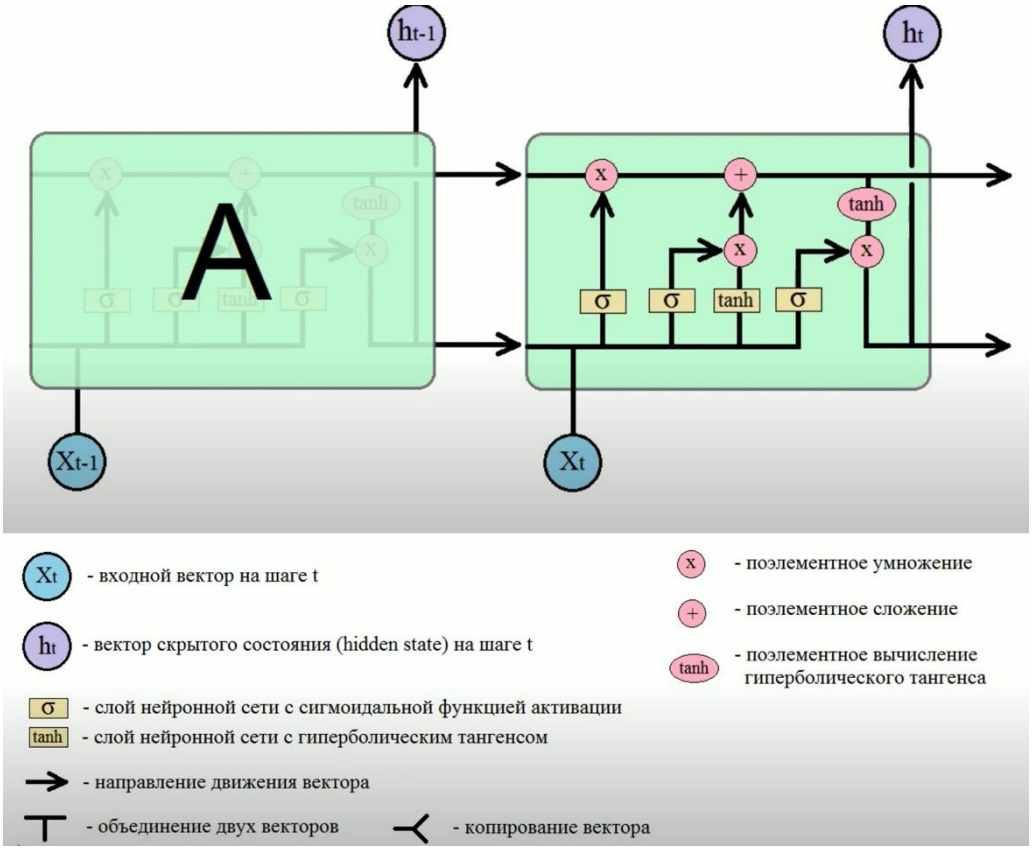

Рис. 11. Блок LSTM.

Существенным отличием данной архитектуры от простой рекуррентной сети является наличие верхнего канала, позволяющего сохранять важный долгосрочный контекст. Благодаря ее наличию у нас есть возможность передавать нужную долгосрочную информацию по рекурсии. Рассмотрим более подробно каждый слой блока LSTM. Пусть в момент времени t на слой с функцией активации σ (формулу см. в приложении 2) подается вектор скрытого состояния и вектор признаков объекта:

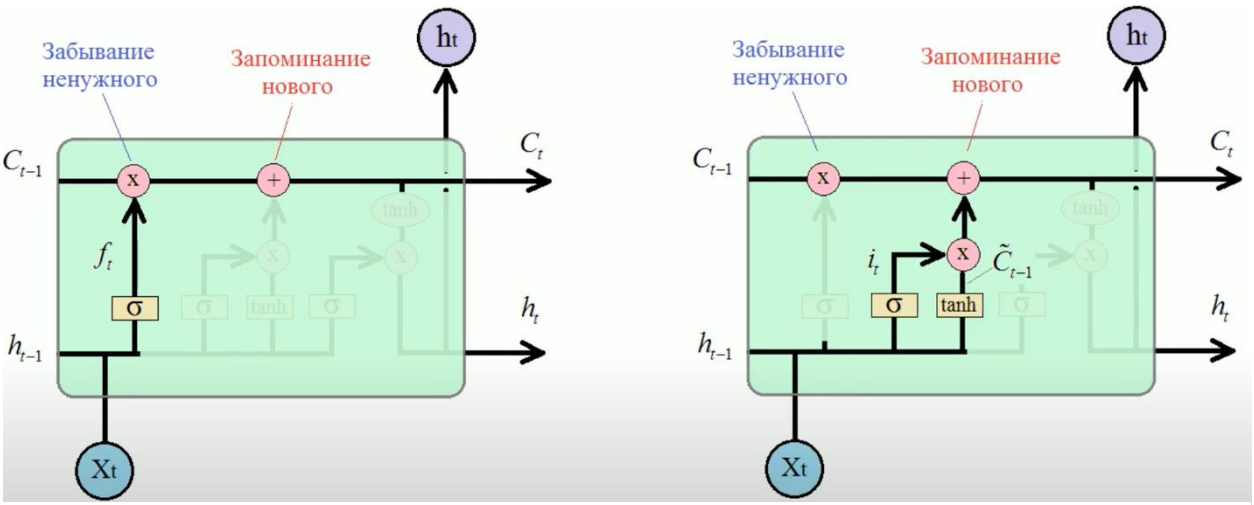

Рис. 13. Принцип работы блока LSTM.

Тогда можно записать, что $f_t = \sigma(W_t * [h_{t-1}, X_t] + b_f)$, $f_t \in [0; 1]$. При поэлементном перемножении с вектором C_{t-1} у нас будут уменьшатся ненужные величины вектора долгосрочного контекста C_{t-1}. Таким образом осуществляется забывание ненужного.

Аналогично запишем выражения для i_t и $\tilde{C}_{t-1}$:

$$i_t = \sigma(W_p * [h_{t-1}, X_t] + b_i)$$

$$\tilde{C}_{t-1} = \tanh(W_C * [h_{t-1}, X_t] + b_C)$$

После эти вектора поэлементно перемножаются, и по той же логике из вектора добавочного контекста будет удаляться ненужная для долгосрочного запоминания информация. Так в блоке меняется запоминающаяся информация от итерации к итерации. Последняя часть блока формирует выходное скрытое состояние h_t.

LSTM-сеть применяется в робототехнике, для анализа временных рядов, для распознавания речи и так далее.

2.3 Архитектура GRU

Архитектура LSTM имеет один существенный недостаток – большое число весовых коэффициентов. Из-за этого процесс обучения таких сетей занимает много времени и памяти компьютера. Поэтому в 2014 году предложили упрощение LSTM, которое называется управляемым рекуррентным блоком **GRU** (Gated Recurrent Units). По эффективности данная модель сравнима с LSTM сетью.

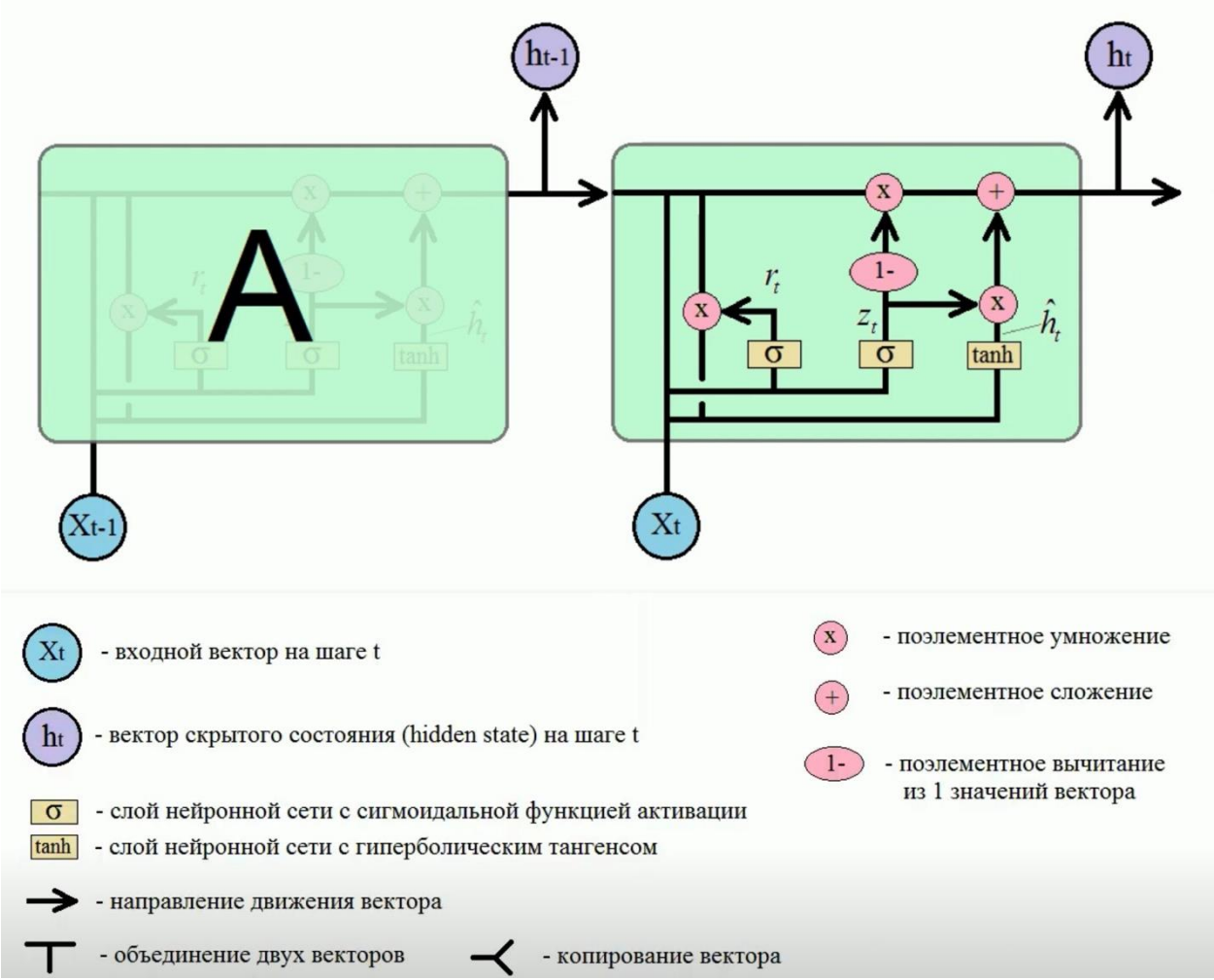

Рис. 14. Блок GRU.

$$z_t = \sigma(W_z * [h_{t-1}, X_t] + b_z), z_t \in [0; 1]$$

$$r_t = \sigma(W_r * [h_{t-1}, X_t] + b_r)$$

$$\hat{h}_t = \tanh(W_{xh} * x_t + W_{hh} * (r_t \otimes h_{t-1}) + b_h).$$

Как можно заметить, главным отличием от LSTM является отсутствие одного выхода, отвечающего за долгосрочный контекст. Эту функцию теперь

выполняет скрытое состояние h_t. В этой модели так же реализуется забывание ненужного и запоминание нового.

$$h_t = h_{t-1} \otimes (1 - z_t) + \hat{h}_t \otimes z_t.$$

W_z в ходе обучения настраиваются так, чтобы сеть знала, что забыть, а что оставить. Противоположная к (1-z_t) информация используется как маркер для добавления нового в соседней ветке. Таким образом, на основе одного и того же вектора формируется информация для забывания и запоминания.

Когда использоваться LSTM, а когда GRU? Все зависит от поставленной задачи. Сначала лучше использовать GRU, потому что она быстрее обучается, а если недостаточно точности, то попробовать применить LSTM.

2.4 Предсказание временных рядов при помощи LSTM и GRU

Прогнозирование временных рядов – это задача, во многом схожая с регрессией. Обладая упорядоченным по времени рядом значений, нам нужно понять, какие значения будут идти в нем дальше. Например, это задачи по предсказанию:

1. Курсов акций;
2. Количества трафика на сайте;
3. Объемов потребления топлива и т.д.

В данной работе мы будем предсказывать значения для **синтетических данных**, то есть для программно сгенерированных данных.

На картинке ниже приведен первый вариант синтетических данных. Обучим модель на 90% этих данных, проверим, как сходится модель на контрольных данных и предскажем 100 точек. У обеих архитектур по 50 скрытых нейронов в одном из двух LSTM/GRU блоках. Длительность обучения – 1000 эпох, размер батча – 32. Сравним MSE для контрольной выборки время обучения для двух моделей.

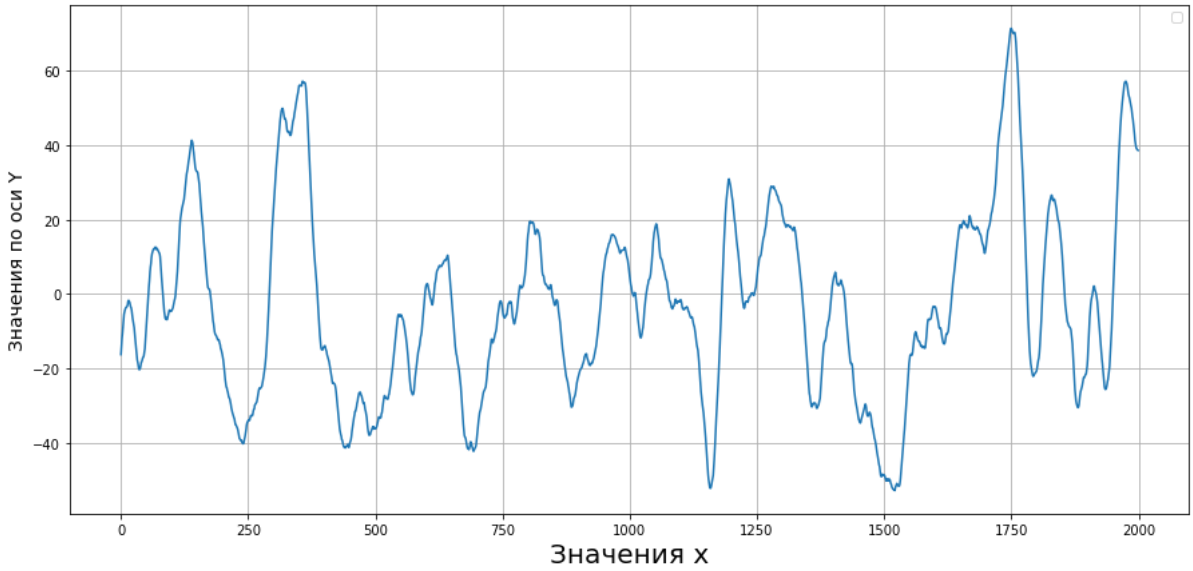

Рис.15.Первый вариант синтетических данных.

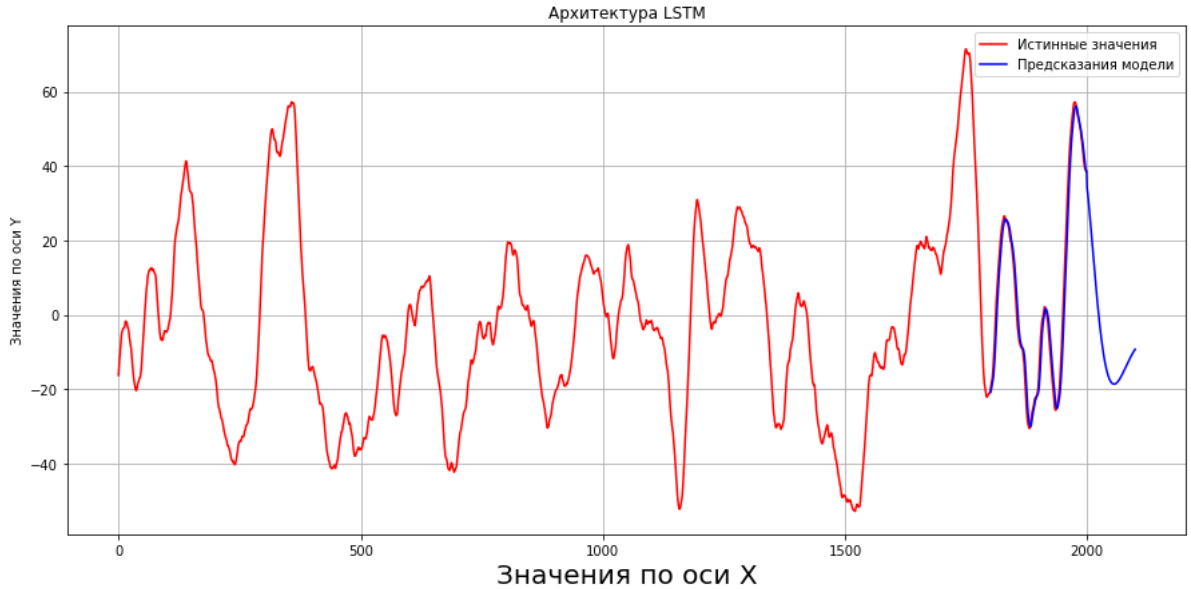

Рис.16. Результат работы LSTM модели.

LSTM:

MSE = 13.4.

Time = 26 minutes 55 seconds.

GRU:

MSE = 15.58

Time = 25 minutes 40 seconds.

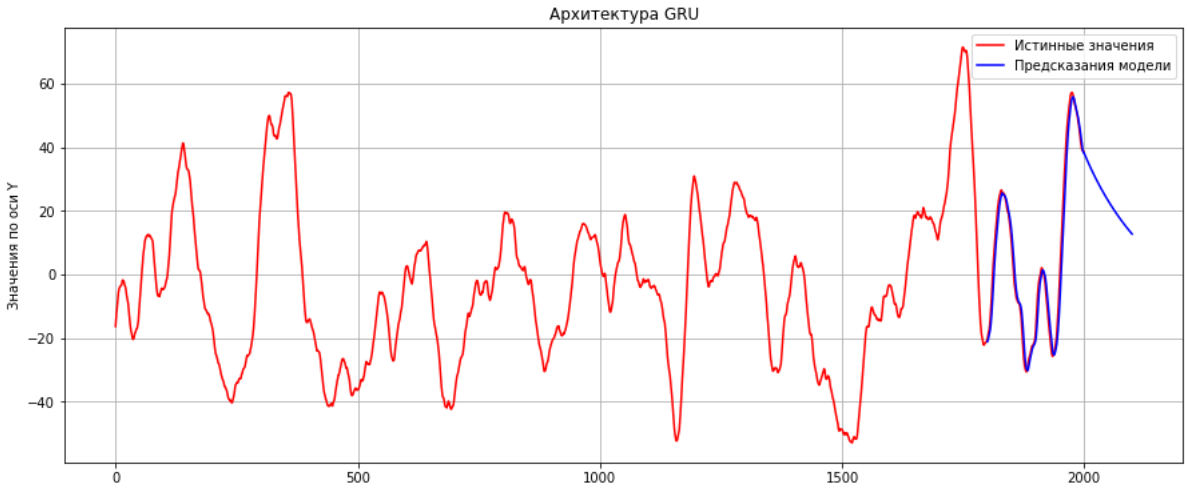

Рис.17. Результат работы GRU модели.

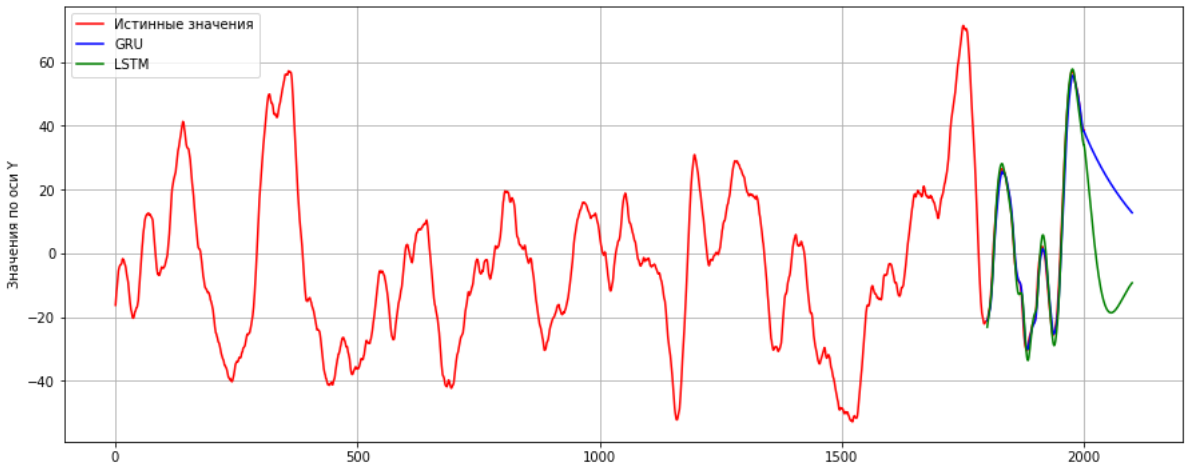

Рис.18. Сравнение двух моделей.

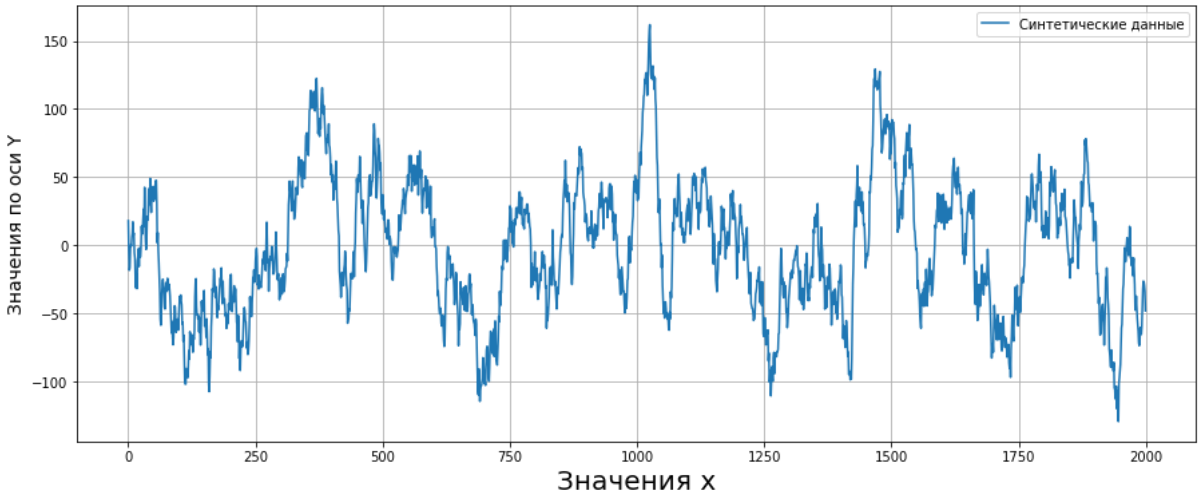

Рис.19. Второй вариант синтетических данных

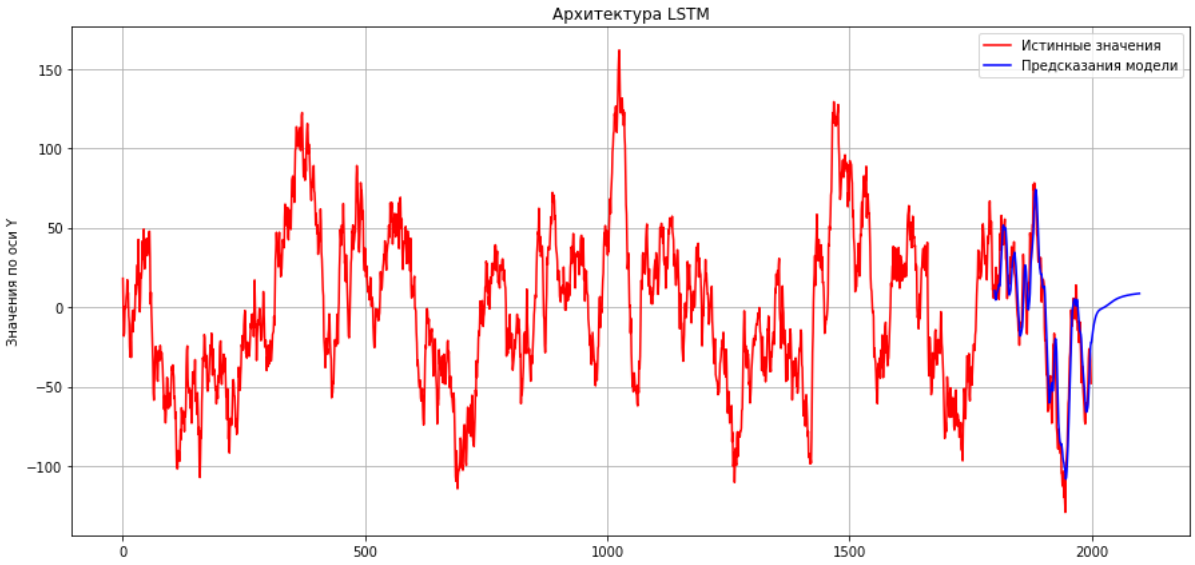

Рис.20. Результат работы LSTM модели.

LSTM:

MSE = 237

Time = 26 minutes 32 seconds.

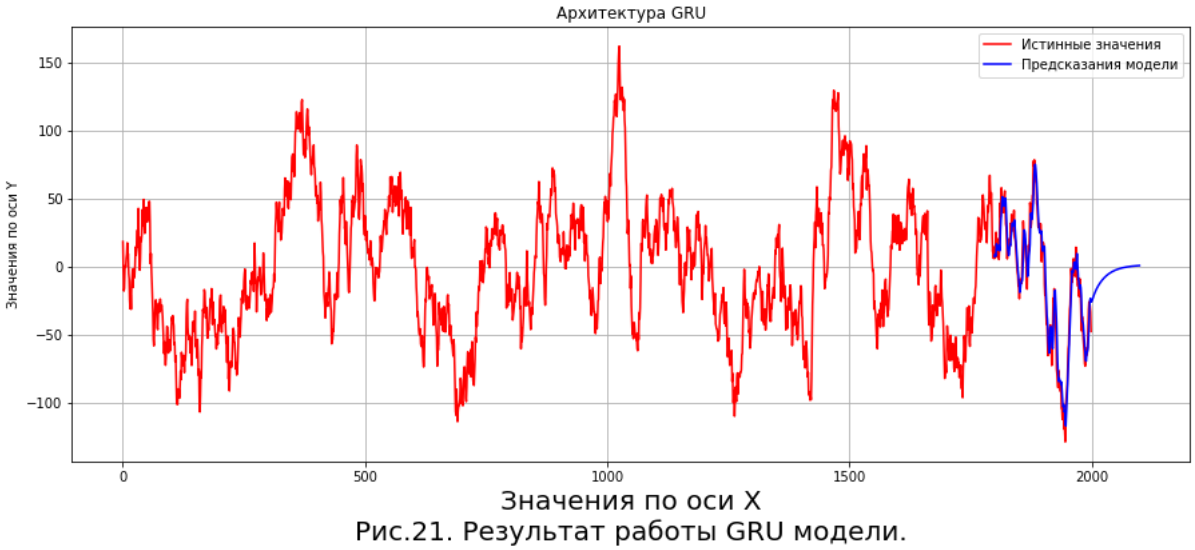

Рис.21. Результат работы GRU модели.

GRU:

MSE = 145

Time = 25 minutes 53 seconds.

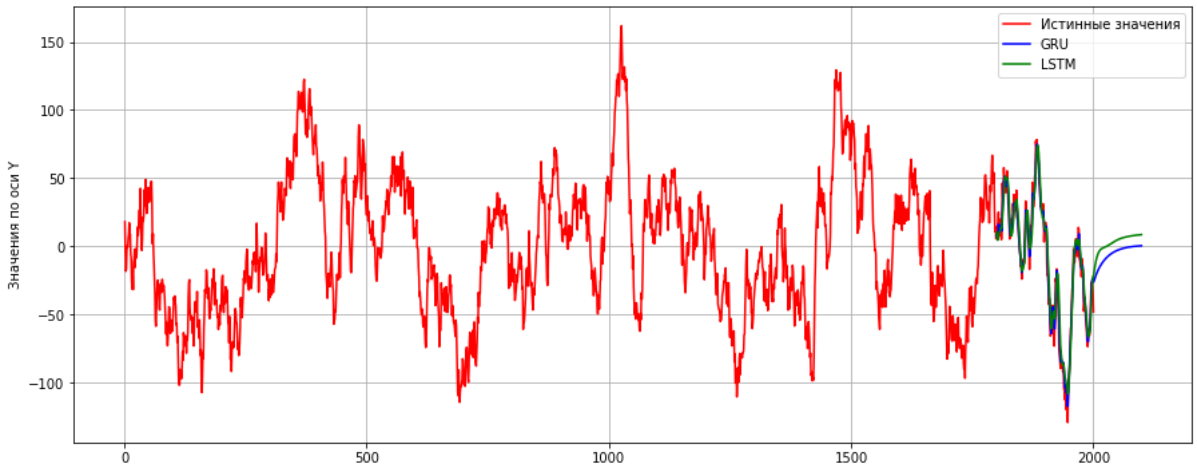

Рис.22. Сравнение двух моделей.

Заключение

В данной работе были изучены простейшие нейронные сети, рекуррентные нейронные сети, LSTM и GRU модели. С их помощью были предсказаны стационарные временные ряды, основанные на синтетических данных. Действительно, архитектура GRU имеет меньшее число настраиваемых параметров и, соответственно, быстрее обучается по сравнению с LSTM моделью. Точность предсказания зависит от вида временного ряда. Для более плавных графиков лучше подходит LSTM модель.

Таким образом, LSTM и GRU модели являются мощным инструментом для анализа временных рядов. Однако их главным минусом является отсутствие понимания, как сети работают внутри: мы не можем узнать, как “мыслит” нейросеть, так как вся информация скрыта в весах и цифрах. Особенно это проявляется при глубоком обучении, когда архитектура сети усложняется в несколько десятков раз.

Список использованных источников

1. https://www.youtube.com/playlist?list=PLA0M1Bcd0w8yv0XGiF1wjerjSZVSrYbjh
2. https://stepik.org/course/50352/syllabus
3. https://neural-university.ru/vocabulary-neural-netwoks#rec118275610
4. https://proproprogs.ru/neural_network
5. https://www.youtube.com/watch?v=A7ecgufuUg8
6. https://www.youtube.com/watch?v=j59jg2TnZC4&t=678s
7. https://www.youtube.com/watch?v=LI94ZkjE_w4
8. Траск Эндрю “Грокаем глубокое обучение”

Приложение 1

Градиент:

$$\frac{\partial E}{\partial W} = \{\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \frac{\partial E}{\partial w_3}, \dots, \frac{\partial E}{\partial w_m}\}$$

Для дифференцирования частных производных по одному параметру воспользуемся правилом дифференцирования сложных функций.

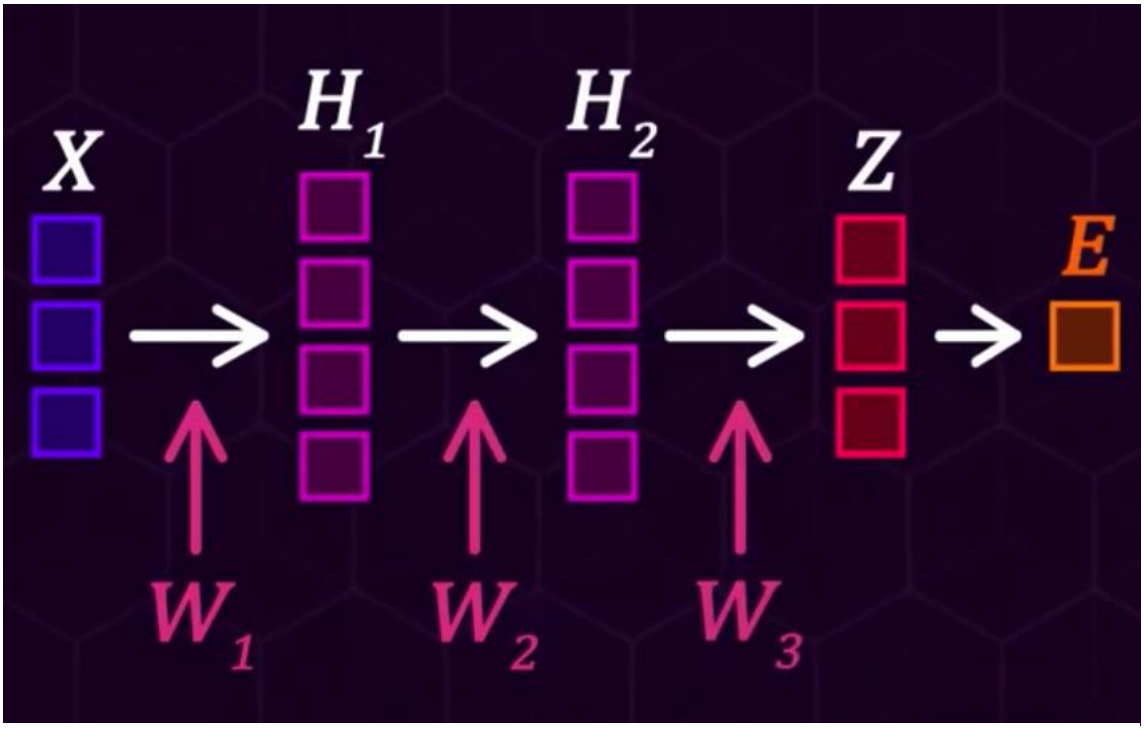

Рис.23. Нейронная сеть с 3 скрытыми слоями.

Пусть градиент, ответственный за параметры первого слоя, равен:

$$\frac{\partial E}{\partial W_1} = \{\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \frac{\partial E}{\partial w_3}, \dots, \frac{\partial E}{\partial w_n}\}$$

где n < m.

Тогда с учетом правила, озвученным выше, получаем:

$$\frac{\partial E}{\partial W_1} = \frac{\partial E}{\partial Z} * \frac{\partial Z}{\partial H_2} * \frac{\partial H_2}{\partial H_1} * \frac{\partial H_1}{\partial W_1}$$

Все частные производные в этом выражении довольно легко вычисляются, так как соответствуют очень простым функциям. Такой алгоритм называется **обратным распространением ошибки.**

Приложение 2

Сигмоидальная функция активации $\sigma(x) = \frac{1}{1+e^{-x}}$

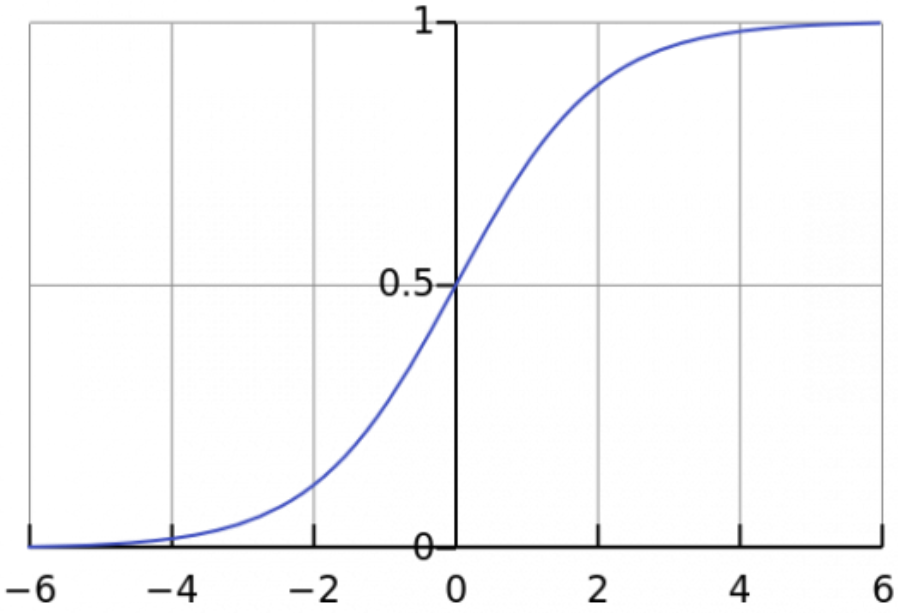

Рис.24. Сигмоидальная функция активации.

Гиперболический тангенс $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

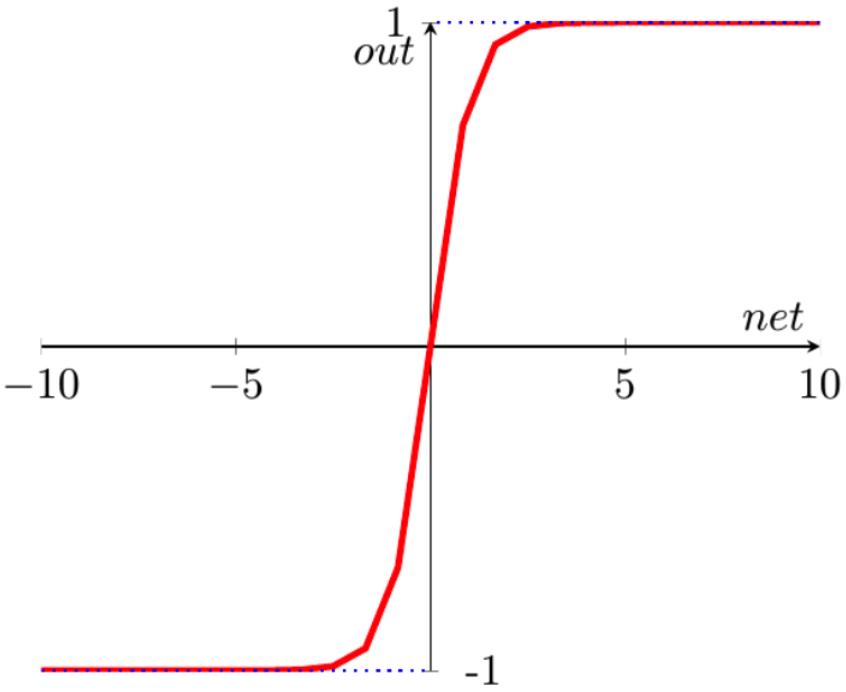

Рис.25. Гиперболический тангенс.