

Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский физико-технический институт»
(национальный исследовательский университет)
Физтех-школа Радиотехники и Компьютерных Технологий
Кафедра радиолокации, управления и информатики

Направление подготовки: 03.03.01 Прикладные математика и физика
Направленность (профиль) подготовки: Радиотехника и компьютерные
технологии

**Метод снижения ошибки определения траектории
сверхзвукового объекта с помощью рекуррентной нейронной
сети, использующей фильтр Калмана**
Бакалаврская работа

Студент:
Хромов Алексей Андреевич

(подпись студента)

Научный руководитель:
Грицык Павел Александрович,
к-т физ.-мат. наук,
заместитель начальника СКБ

(подпись научного руководителя)

Аннотация

Разработан метод и алгоритм, позволяющие оценивать параметры движения объекта, сопровождаемого МФ РЛС, и базирующиеся на нейронной сети. Особенность данного подхода состоит в использовании рекуррентной нейронной сети с долговременной и короткой памятью, и функцией экстраполяции рекурсивного фильтра Калмана. Показано, что новый алгоритм фильтрации имеет высокую точность определения фазового вектора баллистического и гиперзвукового движения, и более низкую среднеквадратичную ошибку фильтрации, по сравнению с классическим фильтром Калмана. В работе обсуждаются методы и принципы нового подхода, предложен конечный вариант программы на языке Python.

Содержание

1 Введение.	5
2 Общие сведения о Фильтре Калмана.	6
2.1 Алгоритм фильтра Калмана.	6
2.2 Математическая модель фильтра Калмана.	8
2.3 Результаты анализа математического моделирования фильтра Калмана. Недостатки метода определения траектории фильтрацией Калмана.	12
3 Общие сведения о нейронных сетях.	13
3.1 Определение нейронной сети.	13
3.2 Линейная модель нейронной сети.	13
3.3 Функции активации нейрона.	16
3.4 Отрицательная обратная связь в обучении нейронной сети.	17
4 Рекуррентная нейронная сеть (RNN).	18
4.1 Причины перехода к рекуррентным нейронным сетям.	18
4.2 Схема рекуррентной нейронной сети.	18
5 Метод снижения ошибки определения траектории с помощью рекуррентной нейронной сети, использующей фильтр Калмана.	20
5.1 Описание метода фильтрации рекуррентной нейронной сетью с кратковременной, долговременной памятью и уравнением экстраполяции Калмана.	20
5.2 Алгоритм рекуррентной нейронной сети с кратковременной, долговременной памятью и уравнением экстраполяции Калмана.	23
6 Математическое моделирование фильтра с рекуррентной нейронной сетью.	24
6.1 Фильтрация траекторий баллистических измерений рекуррентной нейронной сетью.	24
6.1.1 Обучение рекуррентной нейронной сети.	24
6.1.2 Проверка модели рекуррентной нейронной сети на контрольных данных.	26
6.1.3 Сравнение результатов фильтраций нейронной сетью и фильтром Калмана.	26
6.2 Фильтрация траекторий гиперзвуковых измерений рекуррентной нейронной сетью.	30
6.2.1 Обучение рекуррентной нейронной сети.	31

6.2.2	Проверка модели рекуррентной нейронной сети на контрольных данных.	32
6.2.3	Сравнение результатов фильтраций нейронной сетью и фильтром Калмана.	32
7	Заключение.	37
8	Список литературы	38

1 Введение.

Современные способы фильтрации предполагают использование классических линеаризованных фильтров или гребенки фильтров с внешним, ручным подбором коэффициентов, основанным на свойствах конкретного объекта обработки. У данного метода имеется множество преимуществ, включающих в себя явное аналитическое решение, точность и быстродействие. Однако, необходимость подбора коэффициентов под данные и желание увеличить точность фильтра с появлением новых вычислительных возможностей, улучшающих скорость обработки данных, заставляет искать иное решение в анализирование сигналов.

Огромное развитие получило направление нейронных сетей в сфере информационных технологий в эффективной обработке и анализе больших данных. Развитие было вызвано появлением новых возможностей в вычислительной мощности и накоплением достаточного объема информации для обработки в современном аппарате нейронных сетей. Поэтому представляется очень перспективным внедрение нейронных сетей в фильтрацию сигналов, чему и посвящена данная работа.

В квалификационной работе обсуждается теория, связанная с фильтром Калмана, и со структурой рекуррентной нейронной сети. Обговариваются причины выбора данной архитектуры, и как именно сеть должна способствовать в обработке сигнала. Основное изложение практического материала, включая моделирование всех процессов, будет вестись на высокоуровневом языке программирования Python.

Целью работы является разработка метода и, реализующего его, алгоритма фильтрации, базирующегося на рекуррентной нейронной сети, с более высокой точностью, чем фильтр Калмана.

Для достижения поставленной цели, необходимо решить задачи. Проанализировать оценочные значения и потери фильтрации Калмана. Проанализировать, предложенный в квалификационной работе, метод и алгоритм нейронной сети, определяющий оценочное значение баллистической и гиперзвуковой траектории. Предложить по методу и, реализующему его, алгоритму математическую модель определение оценки траектории. Сравнить результаты фильтраций и предсказания баллистических и гиперзвуковых траекторий математической моделью нейронной сети и фильтром Калмана. Задача разработки метода снижения ошибки фильтрации является безусловно ключевой для систем сопровождения и наведения, так как оценка точных фазовых характеристик движения цели является главным в РЭС. Увеличение точности в обработке способствует дальнейшему развитию и достижению новых перспектив в области обороны страны.

2 Общие сведения о Фильтре Калмана.

2.1 Алгоритм фильтра Калмана.

Фильтр Калмана используют для фильтрации зашумленных данных, вид шума которого обычно считается известным (белый шум) [1]. Предположив, что на k -ом шаге уже найдено отфильтрованное значение с РЭС $\mathbf{x}_k^{\text{opt}}$, которое хорошо приближает координату к истинной. Зная физическую природу источника сигнала, например, баллистический снаряд, можно теоретически описать процесс модели. Вводя трехмерную систему координат, для \mathbf{x} можно написать матрицу изменения, учитывая, что источник имеет скорость и ускорение:

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}; \quad \mathbf{F}_{3 \times 3} = \begin{bmatrix} 1 & \Delta t & \Delta t^2/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}. \quad (1)$$

Дополняя \mathbf{x} до 9-мерного вектора, включающего в себя изменения по \vec{y} и \vec{z} , и матрицу перехода до размеров 9×9 , где на диагональных клетках будут расположены блоки вида $\mathbf{F}_{3 \times 3}$ (трехмерного случая, описанного выше), а на недиагональных клетках - нулевые блоки:

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_{3 \times 3} & 0 & 0 \\ 0 & \mathbf{F}_{3 \times 3} & 0 \\ 0 & 0 & \mathbf{F}_{3 \times 3} \end{bmatrix}_{9 \times 9}; \quad (2)$$

Получим девятимерную модель. С её помощью можно экстраполировать следующий фазовый вектор:

$$\bar{\mathbf{x}} = \mathbf{F}\mathbf{x}_k^{\text{opt}}.$$

Добавим динамические входы \mathbf{B} и \mathbf{u} , для коррекции поведения модели:

$$\bar{\mathbf{x}} = \mathbf{F}\mathbf{x}_k^{\text{opt}} + \mathbf{B}\mathbf{u}. \quad (3)$$

При расчете, полученное оценочное значение $\mathbf{F}\mathbf{x}_k^{\text{opt}}$ имеет ошибку, определяемую множеством шумовых факторов. Закон распределения случайных величин может быть неизвестным, но известны дисперсии результатов фильтрации. Пусть \mathbf{P} ковариационная матрица 9×9 с ошибками оценочного фазового вектора $\mathbf{F}\mathbf{x}_k^{\text{opt}}$. Тогда ковариационная матрица для предсказанного события будет считаться:

$$\bar{\mathbf{P}} = \mathbf{F}\mathbf{P}\mathbf{F}^T + \mathbf{Q}; \quad (4)$$

где матрица \mathbf{Q} процесс ковариации, связанный с наличием шума, который описывает случайный характер эволюции системы. Она обуславливает уровень доверия фильтра экстраполированным или измеренным данным [1].

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{3 \times 3} & 0 & 0 \\ 0 & \mathbf{Q}_{3 \times 3} & 0 \\ 0 & 0 & \mathbf{Q}_{3 \times 3} \end{bmatrix}_{9 \times 9}; \quad (5)$$

$$\mathbf{Q}_{3 \times 3} = \int_0^{\Delta t} \mathbf{F}(t) \mathbf{Q}_c \mathbf{F}^\top(t) dt = \begin{bmatrix} \Delta t^5/20 & \Delta t^4/8 & \Delta t^3/6 \\ \Delta t^4/8 & \Delta t^3/3 & \Delta t^2/2 \\ \Delta t^3/6 & \Delta t^2/2 & \Delta t \end{bmatrix} \cdot Q_{\text{spector_noise}};$$

где

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t & \Delta t^2/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix};$$

$$\mathbf{Q}_c = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot Q_s;$$

$$Q_s = Q_{\text{spector_noise}}.$$

Еще не получая измеренного фазового вектора, можно предположить, что на шаге $k + 1$ система эволюционирует согласно законам (3), (4), и измерение окажется близким к $\bar{\mathbf{x}}$.

Идея алгоритма Калмана состоит в оптимальном выборе между показанием с измерителя \mathbf{z} и предсказанием $\bar{\mathbf{x}}$, чтобы получить наилучшее приближение к истинной координате \mathbf{x}_{k+1} . Измеренному фазовому вектору \mathbf{z} мы дадим веса \mathbf{K} , а предсказанному значению $\bar{\mathbf{x}}$ веса $(1 - \mathbf{K})[2]$:

$$\mathbf{y} = \mathbf{z} - \mathbf{H}\bar{\mathbf{x}};$$

$$\mathbf{K} = \bar{\mathbf{P}} \mathbf{H}^\top (\mathbf{H} \bar{\mathbf{P}} \mathbf{H}^\top + \mathbf{R})^{-1}; \quad (6)$$

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{K}\mathbf{y}; \quad (7)$$

$$\mathbf{P} = (\mathbf{I} - \mathbf{K}\mathbf{H})\bar{\mathbf{P}}; \quad (8)$$

где \mathbf{H} измеряющая функция, \mathbf{z} , \mathbf{R} это среднее значение и ковариация шума, \mathbf{y} и \mathbf{K} это невязка и весовая матрица Калмана.

Задача фильтрации — это стремление получить наиболее близкое значение к реальной координате, в отличие от задачи сглаживания. И тут важно отметить, что решается данная проблема **минимизацией среднего значения от квадрата ошибки**¹. Так как в основном при использовании фильтра Калмана данные линеаризуют, представляя траекторию линейной в промежутке итерационного шага, то будем считать фильтр Калмана **линейной моделью**. При фильтрации алгоритмом Калмана в данной работе данные будут считаться линейными, и определение фильтра Калмана, как линейной модели, можно считать справедливым.

2.2 Математическая модель фильтра Калмана.

Выпишем целиком ещё раз основной алгоритм фильтра Калмана, который мы обсуждали в предыдущей главе "Алгоритм фильтра Калмана"(формулы (3), (4), (7), (8)):

1. Этап экстраполяции:

- $\bar{x} = Fx + Bu;$
- $\bar{P} = FP F^T + Q;$

2. Этап коррекции:

- $y = z - H\bar{x};$
- $K = \bar{P}H^T(H\bar{P}H^T + R)^{-1};$
- $x = \bar{x} + Ky;$
- $P = (I - KH)\bar{P};$

В будущей главе "Описание метода фильтрации рекуррентной нейронной сетью с кратковременной, долговременной памятью и уравнением экстраполяции Калмана" этап экстраполяции(предсказания) данных, будет использоваться в новом алгоритме фильтрации данных.

Смоделируем траекторию свободного полета тела в плоскости XY , с белым шумом и проверим работу нашего фильтра. Параметры моделирования:

- Скорость по оси X равна 25.0 м/с;
- Скорость по оси Y равна 25.0 м/с;
- Частота измерений 100 Гц;
- Параметры белого шума по радиусу до тела: ошибка $\sigma_r = 1,6$ м, нулевое математическое ожидание 0.

¹Далее будем называть операцию или функцию среднего значения от квадрата ошибки MSE - mean squared error.

- Параметры белого шума по углу до тела: ошибка $\sigma_{angl} = 0,22$ градуса, нулевое математическое ожидание 0.

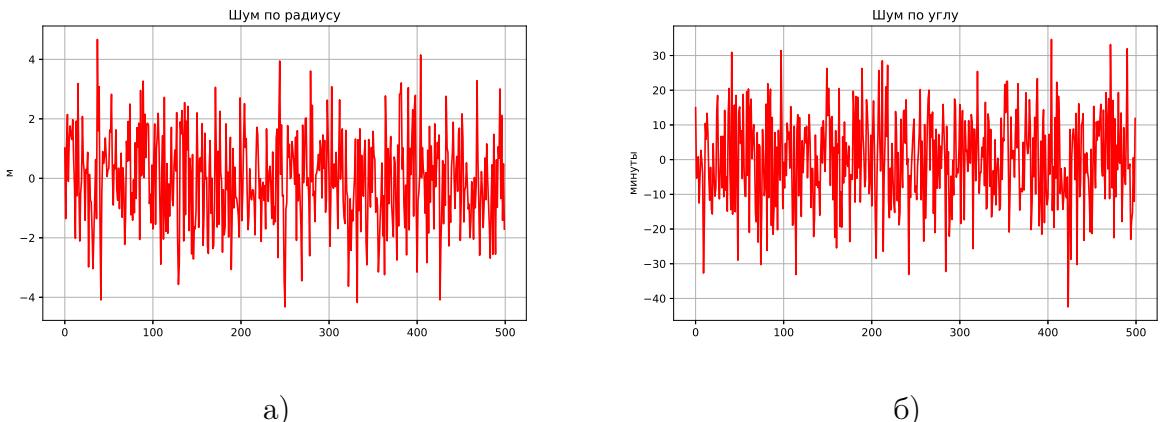


Рис. 1: Графики шума от номера измерения а) по радиусу ; б) по углу.

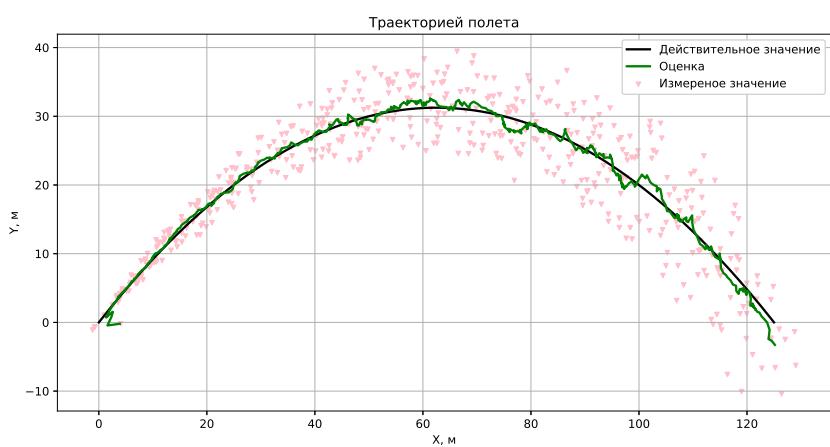


Рис. 2: Полёт тела в плоскости XY

Качественно из рис. 2 видно, что фильтр достаточно точно обрабатывает данные, приближая их к истинным. Теперь выведем графики ошибок: *измеренное значение - истинное значение и оценочное значение - истинное значение*(рис.3 и рис.4).

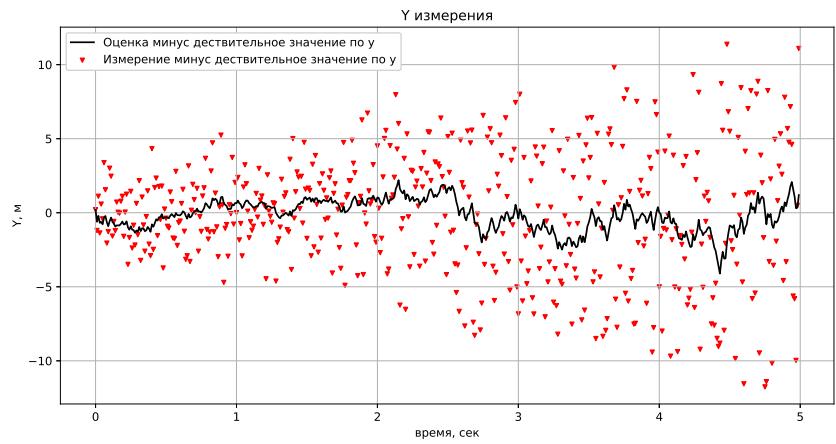


Рис. 3: Отклонение по Y

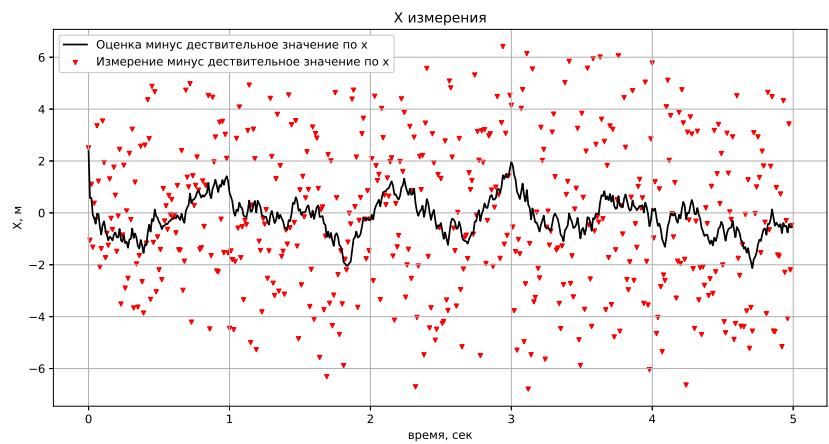


Рис. 4: Отклонение по X

Из графиков видно численно насколько точно воспроизводится фазовый вектор из полученных измерений. Выведем зависимости диагональных элементов в матрице ковариации P , соответствующих ошибкам по координатам X (рис. 5) и Y (рис. 6):

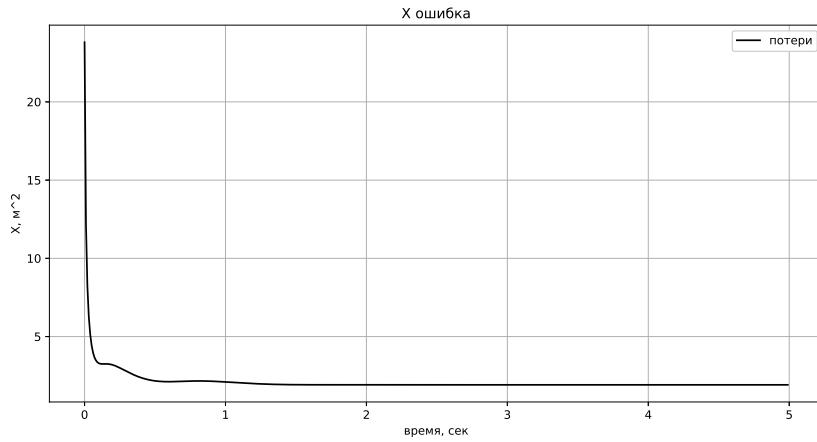


Рис. 5: Среднеквадратичная ошибка по X

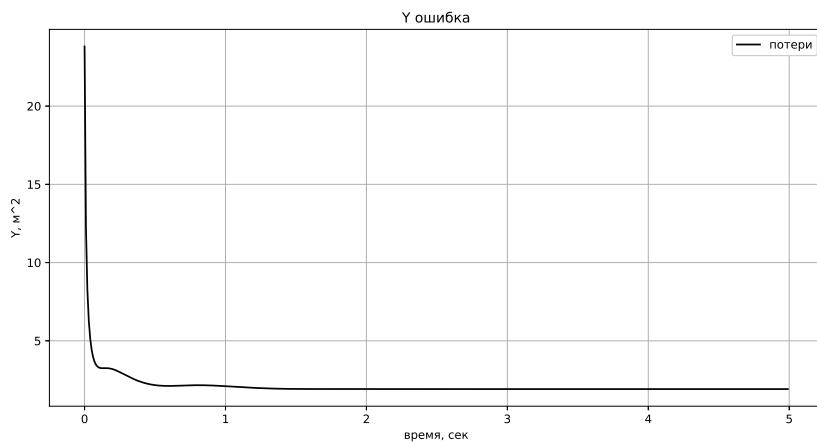


Рис. 6: Среднеквадратичная ошибка по Y

На графиках 5 и 6 видно, что ошибка определения траектории полёта падает, т.е. В процессе фильтрации алгоритм Калмана адаптируется к объекту обработки, что увеличивает точность экстраполяции значений и делает фильтр менее подверженному внешним шумовым факторам.

В следующих главах сравнение нового алгоритма фильтрации будет вестись с построенной в данной главе моделью фильтра Калмана. Будет изучаться реакция и основные характеристики обеих алгоритмов на траектории баллистических и гиперзвуковых объектов.

Теперь же обсудим основные выводы по анализу и моделированию фильтра Калмана рассматриваемого до сих пор.

2.3 Результаты анализа математического моделирования фильтра Калмана. Недостатки метода определения траектории фильтрацией Калмана.

Сгруппируем важные свойства и выводы фильтра Калмана, которые необходимы для дальнейших рассуждений:

- Фильтр Калмана главным образом применяется для линеаризованных данных. Следовательно, можно считать его линейной моделью.
- В алгоритме Калмана решается задача по минимизации среднеквадратичного отклонения между оцененным и истинным значениями;
- Существует аналитическое решение из задачи по минимизации среднеквадратичного отклонения по поиску весовых коэффициентов в матрице K ;
- Высокое приближение фильтрованных данных к истинным.

Фильтр Калмана допускает линейность обрабатываемых данных, и требует ручной настройки - задании внешних параметров обработки, что делает алгоритм ненадёжным при неизвестных характеристиках фильтруемых данных. А в совокупности с необходимостью на каждом шаге искать новую весовую матрицу K путём вычисления обратной матрицы (формула (6)), алгоритм затрудняет поиск и разработку аппаратно-программных решений по увеличению скорости фильтрации. По этим причинам возникает потребность, в новых алгоритмах обработки траекторий, которые могли бы самостоятельно адаптироваться к шумовым характеристикам данных, учитывая нелинейность самих траекторий, имели бы преимущество по оптимизации в вычислительной сложности.

Одним из новых подходов в решении задачи определения траекторий полётов, который рассматривается в работе, является применение нейронных сетей. Приступим к обсуждению основных принципов и преимуществ использования нейронных сетей.

3 Общие сведения о нейронных сетях.

3.1 Определение нейронной сети.

Человеку и многим живым существам в каждый момент приходится решать задачи по распознаванию, классификации и принятию решения, а также, используя результаты последствий принятых действий, обучаться. Нейросетевой подход возник в результате попыток моделирования биологического алгоритма принятий решения – мозга.

Впервые понятие нейронных сетей были введены и разработаны в алгоритмах У. Маккалока и У. Питтса в 1943г. **Нейронная сеть** (NN – англ. neural network) или **искусственная нейронная сеть**, ИНС – математическая модель, построенная по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма, которая последовательными линейными и нелинейными преобразованиями переводит объект из исходного признакового пространства в промежуточные или целевые(межслойные),а в конце и в итоговое пространство.

Нейронные сети способны в отличие от других математических моделей(линейных моделей, логистических регрессий) распознавать сложные и нелинейные зависимости в структурированных данных (траектории движения являются последовательными и структурированными данными). Все потому, что основной задачей нейронных сетей является сначала преобразование данных в новое информативное пространство, а затем поиск решения на новом пространстве для поставленных условий. Возможностью нейросетевого подхода стало появление и накопление достаточно большого количества данных, на которых возможно обучение нейронных сетей.

Важными элементами в нейронных сетях являются линейная модель, нелинейные преобразования(функции активации) и метод обратного распространения ошибки, речь о которых будет в следующих главах.

Перед обсуждением структуры рекуррентных нейронных сетей, далее именуемых RNN², рассмотрим более простые архитектуры и другие важные понятия, необходимые в RNN.

Основой любой NN³ является линейная модель.

3.2 Линейная модель нейронной сети.

Линейная модель - это базисная конструкция, которая по набору признаков принимает решение об принадлежности или свойствах объекта, которому они принадлежат.

$$Y = X_1 + X_2 + X_3,$$

²RNN (англ. recurrent neural network) - рекуррентная нейронная сеть.

³NN (англ. neural network) - нейронная сеть.

Y - решение системы, X_i - категориальные признаки⁴.

Линейная модель решает задачи:

- Классификации - формализованная задача, в которой имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно к каким классам они относятся. Это множество называется выборкой. Классовая принадлежность остальных объектов неизвестна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества (работа с дискретными величинами).
- Регрессии - формализованная задача, в которой есть множество объектов, есть функция на нем. Для некоторого подмножества объектов задано значение функции. Нужно научиться предсказывать значение функции на других объектах. Качество выполнения задачи определяется функционалом качества. Например, MSE (работа с непрерывными величинами).

Подробнее о постановках задач классификации, регрессии и других задач в курсе лекций К. В. Воронцова[4]. Мы же рассмотрим задачу линейной регрессии:

$$\mathbb{E}(Y|X) = f(X);$$

или

$$Y = f(X) = \varepsilon;$$

f - функция регрессии, и для линейной модели имеет вид:

$$f_{\omega}(x) = \omega_0 + \sum_{i=1}^p \omega_i x_i \equiv x^T \omega;$$

где $\omega = (\omega_0, \omega_1, \dots, \omega_n)^T$ - веса, $x = (1, x_1, \dots, x_n)^T$ - признаки объектов.

Пусть теперь мы имеем n объектов: (x^i, y^i) $i = \overline{1, n}$, где $y^i \in R$ - метки объекта, $x^i \in R$ - описание объекта.

Тогда матрицы наших объектов:

$$X = [x^1, \dots, x^n]^T, X \in R^{n \times p}; \quad Y = [y^1, \dots, y^n]^T, Y \in R^n.$$

$$f_{\omega}(X) = X\omega = \widehat{Y}; \tag{9}$$

Осталось только выбрать веса для матрицы ω . Введём функцию эмпирической ошибки, как сумму всех наших потерь:

⁴Многие методы предполагают, что все признаки $X \in R$. Однако, в некоторых задачах признаки могут принимать значения из множеств, не совпадающих с множествами вещественных чисел. Такие признаки называются категориальными, факторными или номинальными [3].

$$\text{Empirical risk} = \sum_{\text{by objects}} \text{Loss on object.}$$

И будем её минимизировать по весам ω , то есть:

$$Q(X) = \sum_{i=1}^n L(y^i, f_\omega(x^i)) \longrightarrow \min$$

Таким образом мы сможем, подбирать ω .

В общей сложности для разных задач существуют разные функции потерь. В данной работе (в задачи регрессии) будет использоваться функция среднеквадратичного отклонения:

$$\text{MSE loss: } L(y_t, y_p) = (y_t - y_p)^2. \quad (10)$$

Тогда:

$$Q_{\text{MSE}} = (Y - X\omega)^T(Y - X\omega) \longrightarrow \min;$$

Для минимизации исследуем функцию потерь, взяв производную:

$$\nabla_\omega Q_{\text{MSE}} = -2Y^T X + 2X^T X\omega^T = 0.$$

Откуда матрица весов:

$$\omega^* = (X^T X)^{-1} X^T Y; \quad (11)$$

По Теорема Гаусса — Маркова[4], данная матрица единственна. В данной постановке задачи и в выводе её решения, чётко прослеживается аналогия с линейной моделью фильтра Калмана, откуда можно сделать вывод, что фильтр Калмана - это модель задачи линейной регрессии.

Как видно из формулы 11, решение ω^* не устойчиво, так как матрица $X^T X$ не всегда имеет обратную. Тут следует дополнить задачу и потребовать от неё устойчивости в решении - провести регуляризацию. Добавим к нашей функции ошибки вторую норму матрицы весов, умноженную на квадрат коэффициента λ :

$$\begin{aligned} L_2 &= \|Y - X\omega\|_2^2 + \lambda^2 \|\omega\|_2^2 \\ \omega^* &= (X^T X + \lambda^2 I)^{-1} X^T Y; \end{aligned}$$

Данный метод регуляризации (сдвиг решения) называется L_2 регуляризацией или регуляризацией Тихонова[6]. Существует L_1 регуляризация - регуляризация через манхэттенское расстояние[5].

Описанное выше аналитическое решение включает в себя инверсию матрицы $X^T X$ (или $X^T X + \lambda I$), что довольно дорого с точки зрения вычислительных ресурсов. Сложность инверсии матрицы можно оценить как $O(p^3 + p^2 N)$. Это приводит нас к

итеративным методам оптимизации, которые являются более эффективными и фактически являются основным подходом к оптимизации в машинном обучении.

Градиентный спуск - один из самых популярных методов оптимизации. Стоит отметить тот факт, что цель минимизации (максимизации) (например, значение функции потерь) должна быть дифференцируема по параметрам модели. Используя градиентный спуск, вектор весов $\mathbf{w}^{(t+1)}$ на шаге $t + 1$ можно выразить в следующем виде:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta_t \nabla Q(\mathbf{w}^{(t)}),$$

где η_t шаг градиента (называемый *learning rate* (англ.)).

Градиент в случае функции потерь MSE принимает следующий вид:

$$\nabla Q(\mathbf{w}) = -2X^T Y + 2X^T X\mathbf{w} = 2X^T(X\mathbf{w} - Y).$$

В этом случае сложность составляет всего $O(pN)$. Чтобы сделать его еще более эффективным, мы будем использовать стохастический градиентный спуск, который вычисляет градиент только по некоторому случайному подмножеству данных (англ. batch) K точек, так что конечная сложность уменьшается до $O(pK)$, где $K \ll N$.

Данный процесс, когда градиентным спуском нашей функции ошибки подбирается матрица весов ω , называется обучением (англ. training).

3.3 Функции активации нейрона.

Линейные модели отлично себя проявляют в классификации или в регрессии **линейных** данных, показывают достойные результаты с данными, которые неплохо аппроксимируются прямой. Например, малый участок зашумленной траектории, обрабатываемый фильтром Калмана.

Однако, большинство данных (траектория движения), плохо или вообще не аппроксимируются прямой, так как имеют другие зависимости (например логарифмические), поэтому нам необходимо «уйти» от *нелинейного* признакового пространства в новое пространство (чаще всего линейное), где мы сможем работать с новыми признаками знакомыми способами. Идея очень проста, добавим **нелинейное преобразование**:

1. Входные данные: X ;
2. Линейная модель: $h = X\omega$;
3. Нелинейное преобразование: $\sigma(h)$;
4. MSE;
5. Предсказание.

Нелинейную функцию, в частности σ , называют **функцией активации**. Функций активаций бывает множество видов, приведем в качестве примера пару из них:

- $\sigma(x) = \frac{1}{1+e^{-x}}$, $R \rightarrow (0, 1)$;
- $\text{th}(x)$, $R \rightarrow (-1, 1)$;
- $\ln(1 + e^x)$, $R \rightarrow (0, \infty)$;

Функции активации позволяют нашей сети, переводить признаки в новое удобное для себя пространство, где NN сможет с ними работать или перевести в еще одно признаковое пространство. Множество таких линейных и нелинейных преобразований называются слоями. Присутствие функций активации в преобразованиях NN является одной из отличительных черт, почему NN превосходит классические линейные модели в работе с нелинейными данными.

Для работы NN имеет в каждом слое(преобразование) обучаемый параметр или несколько, и требует, чтобы все слои имели производную в точке, для возможности обучаться градиентным спуском. Хорошая нейронная функция должна быть ограниченной, легко дифференцируемой, монотонной (хорошо подходит для выпуклой оптимизации) и простой в вычислительной сложности.

Примечание.

Название и происхождение функций активации уходит в биологию и психологию. Когда люди решили смоделировать поведение *нейронов*, они обнаружили, что множество нейронов передают потенциал одному, и тот, после превышения порогового значения(*активации*), начинает передавать накопившийся потенциал остальным. То есть для работы модели нейронов или нейронной сети требуется функция активации, которая была бы «дифференцируема». Первой такой функцией стала σ .

3.4 Отрицательная обратная связь в обучении нейронной сети.

Скажем пару слов об отрицательной обратной связи в обучении нейронной сети(методе обратного распространения ошибки). Данный метод является приемом вычисления градиента, для обновления весов в многослойной нейронной сети. Главным принципом является вычисление производной в точке сложной функции.

Рассмотрим нейронную сеть из пары слоев:

1. Входные данные: X ;
2. Линейная модель: $h = X\omega$;
3. Нелинейное преобразование: $\sigma(h)$;
4. MSE;

5. Предсказание.

На этапе поиска ошибки(MSE формула (10)) идет обучение нейронной сети, ставится задача минимизации потерь:

$$MSE = L \rightarrow \min.$$

Для этого необходимо посчитать производную L по параметрам обучения–весам, и приравнять к нулю.

$$\frac{\partial L}{\partial \omega} = 0.$$

Но L сложная функция от ω , так как $L(\sigma(\omega))$, тогда производная будет считаться по формуле:

$$\frac{\partial L}{\partial \omega} = \frac{\partial L}{\partial \sigma} \frac{\partial \sigma}{\partial \omega}.$$

Зная численное значение вложенных функций в точке, производная сложной функции ищется произведением найденного значения последних производных($\partial L / \partial \sigma$) на рассчитанное значение следующей производной($\partial \sigma / \partial \omega$). После нахождения необходимой производной обновляем весовые параметры:

$$\omega_{n+1} = \omega_n - \eta \frac{\partial L}{\partial \omega},$$

где η – итерационный шаг градиента.

4 Рекуррентная нейронная сеть (RNN).

4.1 Причины перехода к рекуррентным нейронным сетям.

Чаще всего данные, которые встречаются, и которые необходимо обрабатывать, обладают структурностью. Например, речь, где порядок слов и букв обладает собственной архитектурой и несут важную информационную нагрузку. При обработке тут важно учитывать свойства структурности информации.

Траектория движения имеет последовательность точек, которые привязаны ко времени, это означает, что каждая последующая координата зависит от совокупности предыдущих. То есть при фильтрации новых значений помимо знаний о физической природе объекта можем пользоваться доступным контекстом - уже обработанными точками траектории.

4.2 Схема рекуррентной нейронной сети.

Пусть имеются данные и часть контекста, связанного с ними. Теперь «обогатим» контекст следующими данными, и получим результат обработанных значений в совокупности с предыдущими. Для этого представим свои данные в виде векторов: C - контекст

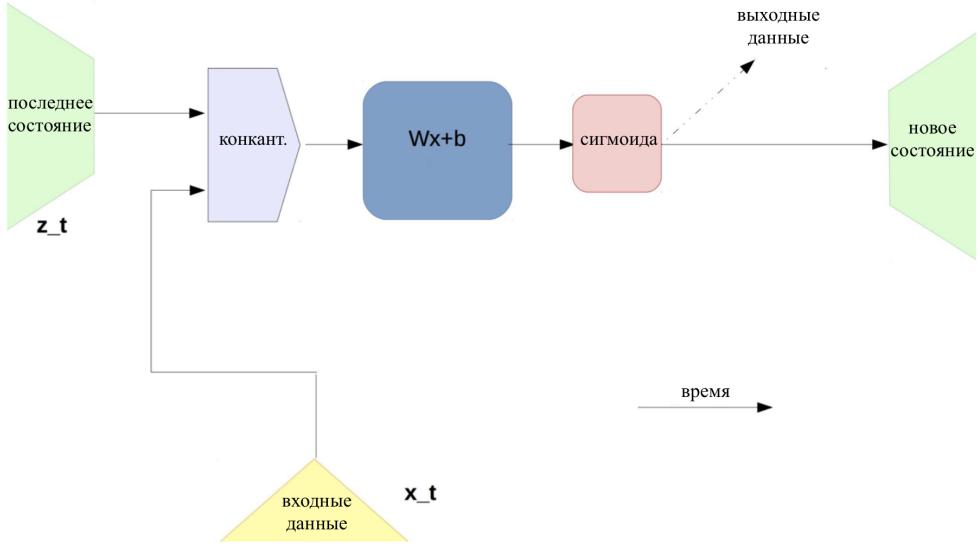


Рис. 7: Схема RNN loop[9].

фиксированной размерности, i - входные данные фиксированной размерности. Для объединения старой и полученной информации конкatenируем вектора C и i в $[C, i]$. Полученные новые данные(нужной размерности) линейно преобразуем: $W \cdot [C, i] + b$. Нейронной сети необходимо иметь нелинейность в своих преобразованиях, поэтому добавим функцию активации после линейного слоя: $\sigma(W \cdot [C, i] + b)$. Получили композицию преобразований, способную называться нейронной сетью, так как в ней имеется нелинейность в преобразованиях, и веса W для обучения. Скажем, что $W \cdot [C, i] + b$ порождает вектор размерности, соответствующей изначальному контексту C , следовательно, получаем обновленный контекст, способный участвовать в дальнейшей циклической обработке данных. Нейронные сети данного типа, называют рекуррентными, а один оборот процесса - петлёй (англ. loop), рис. 7.

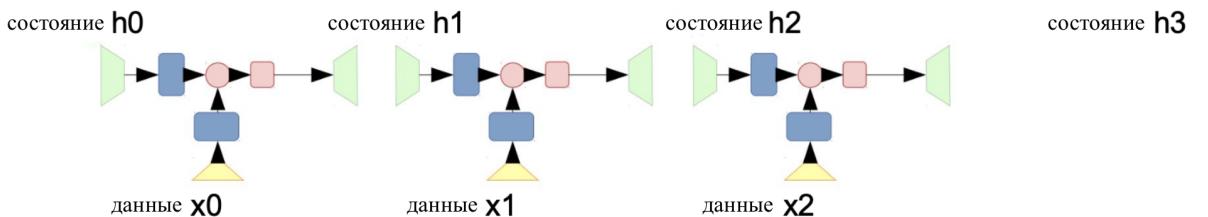


Рис. 8: Простейшая схема RNN[9].

На рис. 8 показана схема циклической работы RNN.

Можно заметить, что первоначальная информация вектора контекста после нескольких циклов добавления новых данных быстро «забывается» из-за фиксированной размерности C , и мы теряем информацию о начальных событиях. Учитывая, что в данных

может присутствовать много зашумленных или ненужных значений, имеет смысл запоминать важную информацию и пропускать остальную - дать нашей сети возможность выбирать что нужно запомнить, а что нет.

5 Метод снижения ошибки определения траектории с помощью рекуррентной нейронной сети, использующей фильтр Калмана.

5.1 Описание метода фильтрации рекуррентной нейронной сетью с кратковременной, долговременной памятью и уравнением экстраполяции Калмана.

LSTM (англ. Long Short-Term Memory) - рекуррентная нейронная сеть с кратковременной и долговременной памятью⁵. Основная структура схемы LSTM на рис. 9.

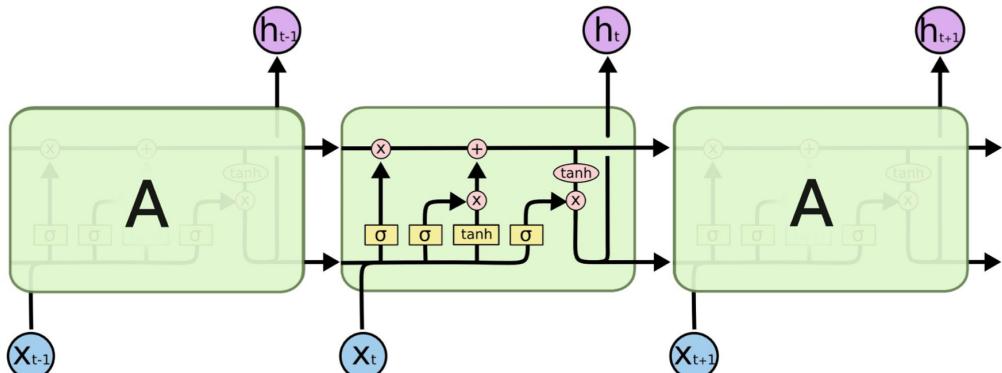


Рис. 9: LSTM[10].

LSTM использует два вектора контекста: - долговременная память, h - кратковременная память. Задача выбора: какую информацию оставить, а какую забыть, является бинарной классификацией. f_t - вектор "забывания" содержит 0, где данные нужно стереть, и 1, где сохранить:

⁵Основной материал по LSTM можно найти в лекции Стэнфорда [10]

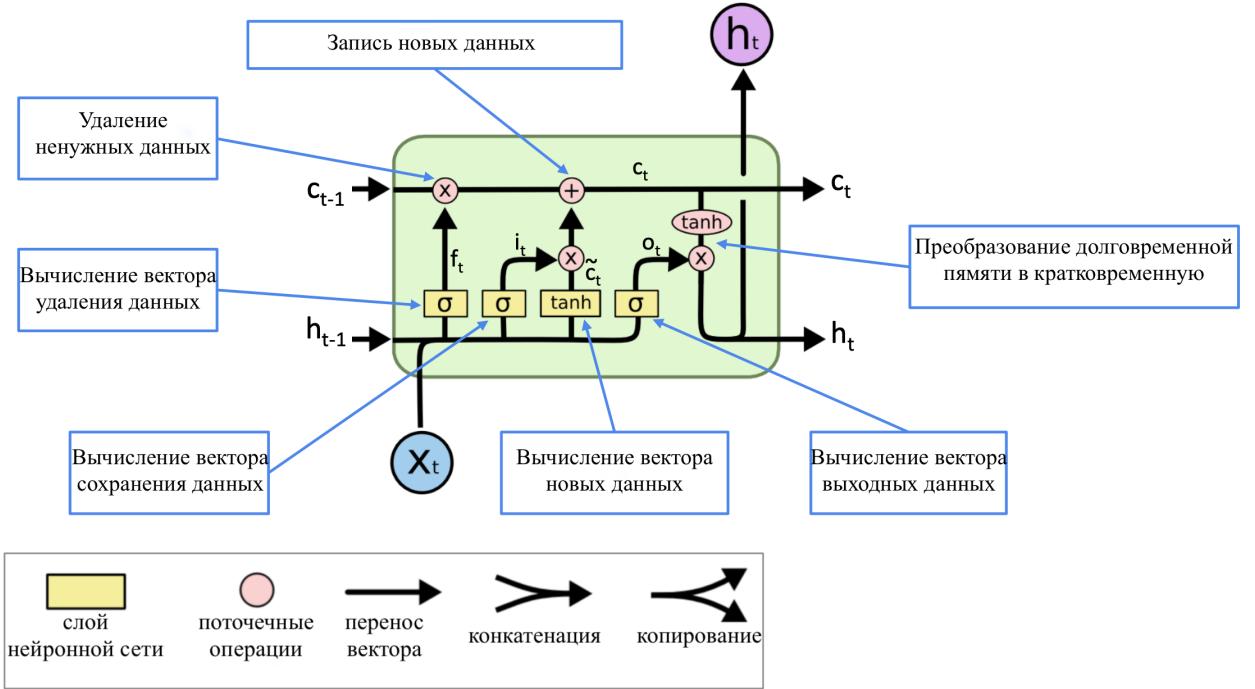


Рис. 10: LSTM loop[10].

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

Функция σ (гл. "Функции активации нейрона") бинарно классифицирует данные $(0, 1)$, после чего совершается поэлементное умножение f_t на C_{t-1} . Теперь нужно запомнить новую информацию для этого - решается задача классификации, пусть i_t - вектор "запоминания" аналогичный f_t :

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

Нужны сами данные \tilde{C}_t для запоминания:

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

Так как теперь решается задача по управлению вектором, где преобразуются и хранятся данные - вектором состояния, который должен быть способен увеличиваться или уменьшаться при добавлении новых данных от внешней функции, то используется \tanh как функция активации. Сигмовидный выход всегда неотрицателен, значения в состоянии будут только увеличиваться. Выход из \tanh может быть положительным или отрицательным, что позволяет увеличивать и уменьшать состояние.

Ограниченнность распределения данных у \tanh в интервале $(-1, 1)$, и их центрированность относительно 0, позволяет \tanh не перегружаться от больших значений, что важно для предотвращения взрыва градиента⁶. С другой стороны, чтобы преодолеть

⁶Взрыв градиента - понятие, когда градиент быстро растёт и вызывает переполнение данных.

проблему исчезающего градиента в рекуррентных нейронных сетях, при расчете новых весовых коэффициентов методом обратного распространения ошибки (гл. "Отрицательная обратная связь в обучении нейронной сети"), нужна функция, вторая производная которой может сохраняться в течение длительного времени, прежде чем превратиться в ноль.

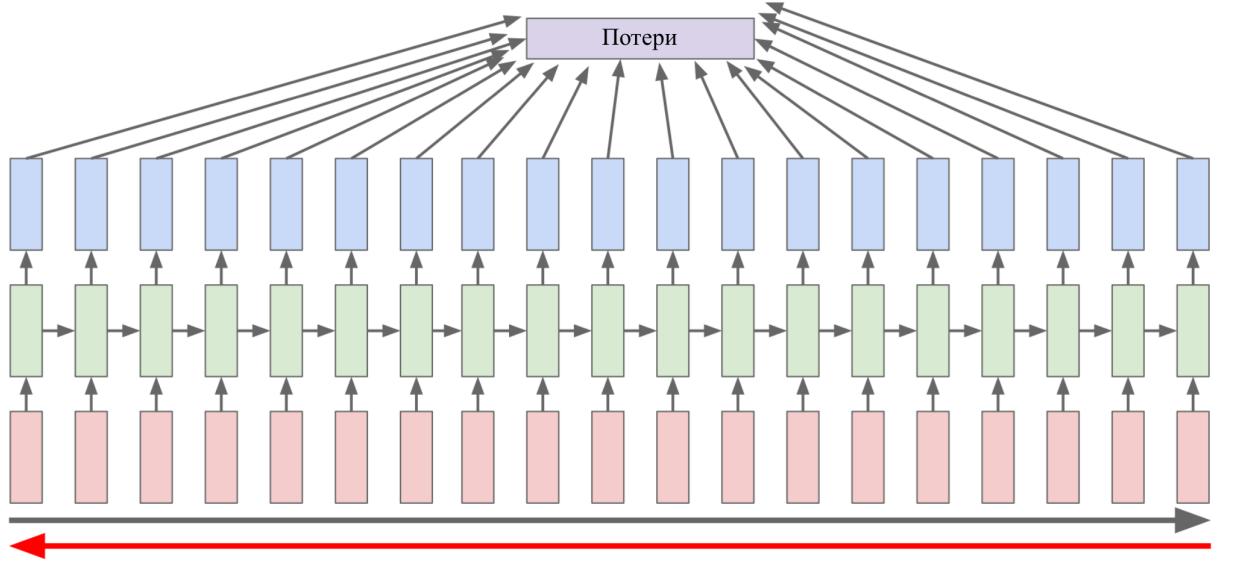


Рис. 11: Обучение во времени.

Обновление долговременной памяти:

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

Изменение краткосрочной памяти, аналогично C_t :

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

$$h_t = o_t \cdot \tanh(C_t).$$

Для решения задачи фильтрации в алгоритме Калмана заменим пункт **Обновления** формулы 7, 8 на LSTM. Сеть будет получать зашумленный сигнал x_{noise_t} , выводить уже отфильтрованное значение h_t . Далее по формулам 3 на основе предыдущего значения фильтрации, пользуясь свойствами физической модели объекта, предсказываем предположительное значение \tilde{x}_t . Конкатенируем $[\tilde{x}_t, h_t]$ и совершаю линейное преобразование, получая вектор размерности координат точки:

$$\tilde{x}_t = Fx_{t-1};$$

$$x_t = W_K \cdot [\tilde{x}_t, h_t] + b_k. \quad (12)$$

Последний линейный слой (формула 12), описанный выше, будет обучаться нейронной сетью, так как является её частью. Подбирать весовые параметры NN будем методом стохастического градиентного спуска от Loss функции 10(фильтрация сигнала - задача регрессии), считая производные слоев[7] рис. 11.

5.2 Алгоритм рекуррентной нейронной сети с кратковременной, долговременной памятью и уравнением экстраполяции Калмана.

Архитектура сети уже обсуждалась в предыдущем разделе ("Описание метода фильтрации рекуррентной нейронной сетью с кратковременной, долговременной памятью и уравнением экстраполяции Калмана"). Практический алгоритм представлен на листинге 1. Нейронная сеть написана на основе фреймворка машинного обучения PyTorch с открытым исходным кодом.

Создаем класс данных(основанный на стандартах Dataset PyTorch), предварительно приводя его в стандартизированный вид фиксированной длины.

```
1 class ModuleRNN(nn.Module)
```

Класс нейронной сети, который содержит в себе главную архитектуру системы. В методе forward описан алгоритм, обсужденный в главе "Описание метода фильтрации рекуррентной нейронной сетью с кратковременной, долговременной памятью и уравнением экстраполяции Калмана":

Листинг 1: ModuleRNN.forward()

```
1 def forward(self, new_data, flash_memory, short_term_memory,
2             long_term_memory, F):
3
4
5     memory = torch.cat([new_data, short_term_memory], dim=-1)
6
7
8     forgetfulness = torch.sigmoid(self.rnn_forget(memory)) #
9         forgetting dataforgetting data
10    conservation = torch.tanh(self.rnn_save(memory)) #the
11        acquisition of new data
12    information = torch.sigmoid(self.rnn_data_selection(memory))
13
14
15    long_term_memory = (forgetfulness * long_term_memory) + (
16        information * conservation)
```

```

11     short_term_memory = torch.sigmoid(self.rnn_quick_overview(
12         memory)) * torch.tanh(long_term_memory)
13
14     with torch.no_grad():
15         flash_memory = self.predict_Kalman(F, flash_memory)
16         flash_memory_grad = flash_memory[:, 0::3]
17
18     predicted_data = self.rnn_prediction(torch.cat([
19         flash_memory_grad, short_term_memory], dim=-1))
20
21     with torch.no_grad():
22         flash_memory[:, 0::3] = predicted_data
23
24
25     return predicted_data, flash_memory, short_term_memory,
26         long_term_memory

```

В коде имеются области `with torch.no_grad()`, отключающие поиск производных для воведших в них операций, так как они не являются дифференцируемыми⁷.

```
1 def rnn_loop()
```

Главный цикл в котором работает RNN.

6 Математическое моделирование фильтра с рекуррентной нейронной сетью.

6.1 Фильтрация траекторий баллистических измерений рекуррентной нейронной сетью.

6.1.1 Обучение рекуррентной нейронной сети.

Приведём процесс обучения и проверка точности(валидации⁸) в виде графиков зависимостей от эпох⁹:

Напомним, что подбираем веса параметров NN методом стохастического гради-

⁷Срезы тензоров используются для оптимизации, так как являются более быстрыми и эффективными операциями чем матричные умножения

⁸Валидация - это проверка данных на соответствие заданным условиям и ограничениям.

⁹Эпоха - время, за которое обрабатываются все данные.

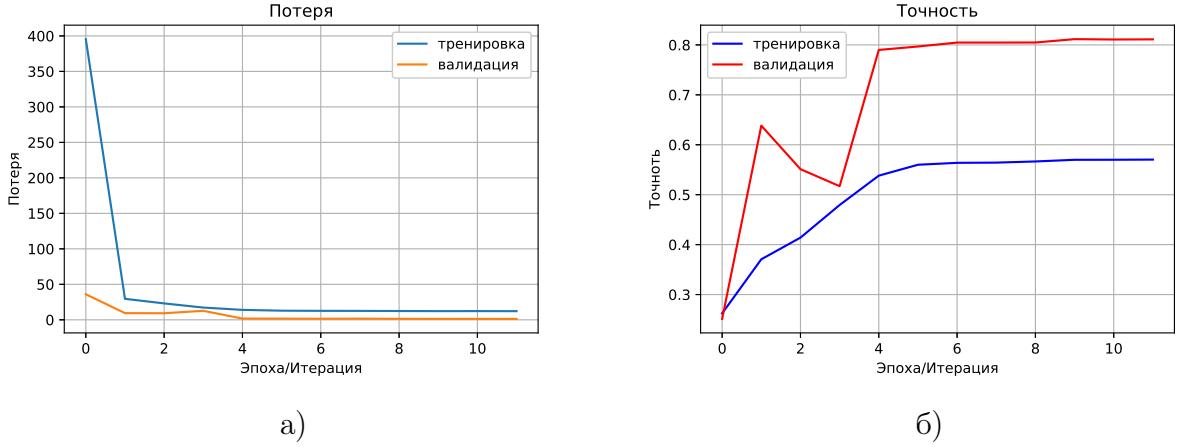


Рис. 12: Процесс обучения и валидации. а) график зависимости средней по данным MSE ошибки от эпохи; б) график зависимости средней по данным точности от эпохи.

ентного спуска от функции потерь MSE ¹⁰(формула 10):

$$\text{MSE loss: } L(y_t, y_p) = (y_t - y_p)^2.$$

Подсчет потерь ведется по формуле:

$$loss = \sqrt{\frac{1}{S} \sum (\hat{\mathbf{x}} - \mathbf{x}_{\text{real}})^2}, \quad (13)$$

где $\hat{\mathbf{x}}$ – вектор оценочных значений фазового вектора, \mathbf{x}_{real} – действительный фазовый вектор, S – количество элементов в фазовом векторе. Сначала находится поэлементная разница двух векторов, потом поэлементное возвведение в квадрат. Элементы полученного вектора скалываются и делятся на их общее количество. Далее идет извлечение корня.

"Точность" осуществляется подсчетом количества точек фильтрованного сигнала, отклонившихся от оригинала не более чем на 5%. То есть сначала вычитается поэлементно из оцененного вектора действительный фазовый вектор. Полученная разница поэлементно делится на действительный вектор, а затем поэлементно логически сравнивается с критерием(D). Получается вектор из единиц и нулей, после суммирования всех элементов вектора, имеем общее число координат, удовлетворивших критерию. Данное число делится на общее число всех элементов. Итоговое значение – это среднее число оцененных координат удовлетворивших критерию.

$$accuracy = \frac{1}{S} \sum \left(\frac{|\hat{\mathbf{x}} - \mathbf{x}_{\text{real}}|}{\mathbf{x}_{\text{real}}} \leq D \right), \quad (14)$$

где $\hat{\mathbf{x}}$ – вектор оценочных значений фазового вектора, \mathbf{x}_{real} – действительный фазовый вектор, $D = 0,05$ – допустимая ошибка 5%, S – количество элементов в фазовом векторе.

¹⁰MSE – функция потерь (англ. loss), среднеквадратичная ошибка.

Вывод: Из падения графиков на MSE и росте на Accuracy видно, что сеть обучается. Точность на валидации чуть выше 81%, это связано с регуляризацией¹¹ модели.

6.1.2 Проверка модели рекуррентной нейронной сети на контрольных данных.

Приведём проверку обученной модели на контрольных данных(тестирование) в виде графиков:

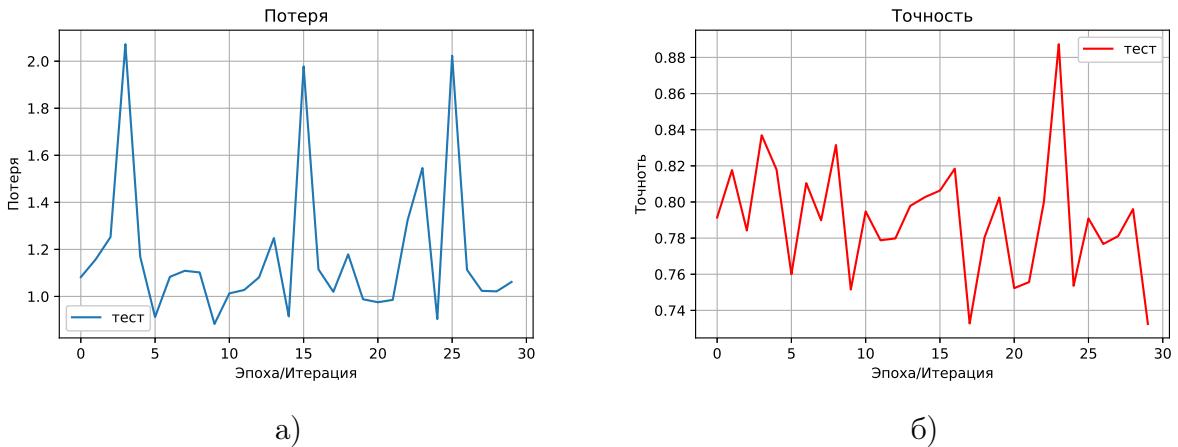


Рис. 13: Процесс тестирования. а) график зависимости средней по данным MSE ошибки от эпохи; б) график зависимости средней по данным точности от эпохи.

Потери и точность считаем по формулам из предыдущей главы (13), (14)(гл. "Обучение рекуррентной нейронной сети") Ошибка в среднем 1, 18 км по траекториям. А точности 5% в среднем достигается в 79, 0% фильтрации.

Вывод: Сеть обучилась на наших данных, но возможен рост характеристик точности при увеличении обучающей выборки и количества весовых параметров.

6.1.3 Сравнение результатов фильтраций нейронной сетью и фильтром Калмана.

На данный момент уже разобраны фильтр Калмана и фильтр, с применением рекуррентной нейронной сети. Сравним результаты фильтраций траекторий полетов баллистических снарядов, уже привычными, функциями (13), (14) (из главы "Обучение рекуррентной нейронной сети"):

$$loss = \sqrt{\frac{1}{S} \sum (\hat{\mathbf{x}} - \mathbf{x}_{\text{real}})^2},$$

$$accuracy = \frac{1}{S} \sum \left(\frac{|\hat{\mathbf{x}} - \mathbf{x}_{\text{real}}|}{\mathbf{x}_{\text{real}}} \leq D \right),$$

¹¹В данном случае говорится об секуляризации Dropout.

где $\hat{\mathbf{x}}$ – вектор оценочных значений фазового вектора, \mathbf{x}_{real} – действительный фазовый вектор, $D = 0,05$ – допустимая ошибка 5%, S – количество элементов в фазовом векторе.

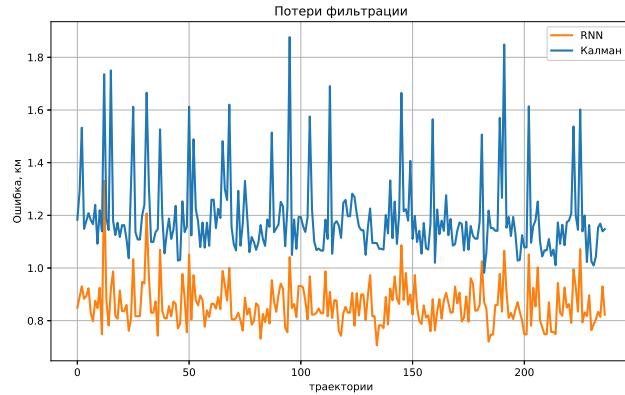


Рис. 14: Средние MSE потери по траекториям

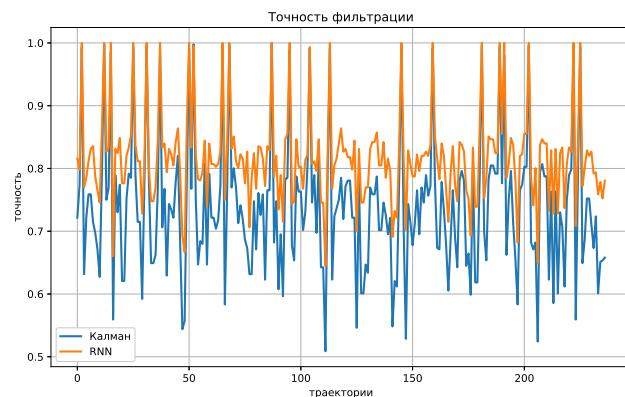


Рис. 15: Средняя точность по траекториям

Средние по траекториям потери RNN фильтра Калмана составляют 0,9 км, тогда как у классического девяти-мерного фильтра они равны 1,2 км (Рис. (14)).

В 5%-ую точность в среднем по траектории попадают 81% результатов фильтрации RNN, и 73% для фильтра Калмана (Рис. (15)).

Для исследования характеристик фильтров рассмотрим их работу не в среднем, а на конкретной произвольной траектории. Будем выводить графики вида: |«сигнал»-«сигнал+шум»| и рядом |«фильтрованный сигнал»-«сигнал+шум»| в зависимости от времени.

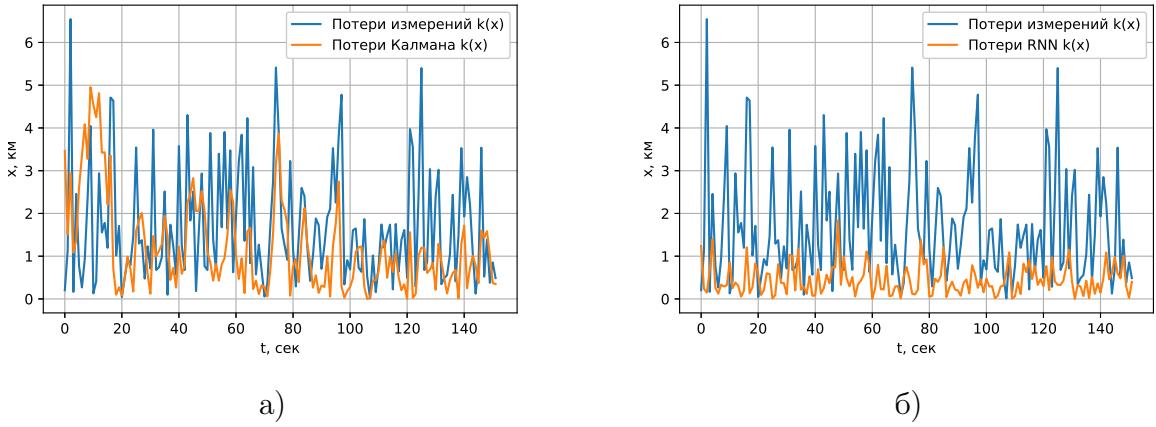


Рис. 16: Графики ошибки по оси x . а) для классического фильтра Калмана; б) для фильтра Калмана с NN.

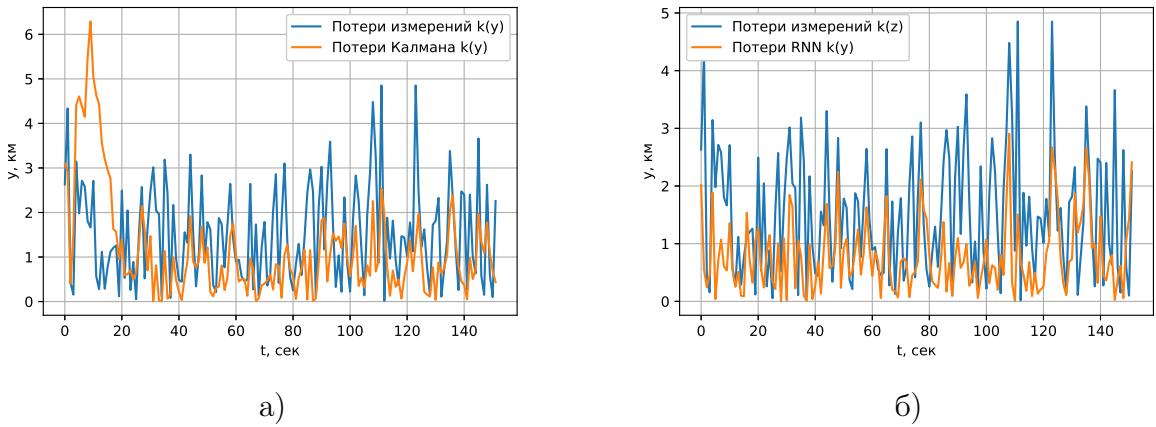


Рис. 17: Графики ошибки по оси y . а) для классического фильтра Калмана; б) для фильтра Калмана с NN.

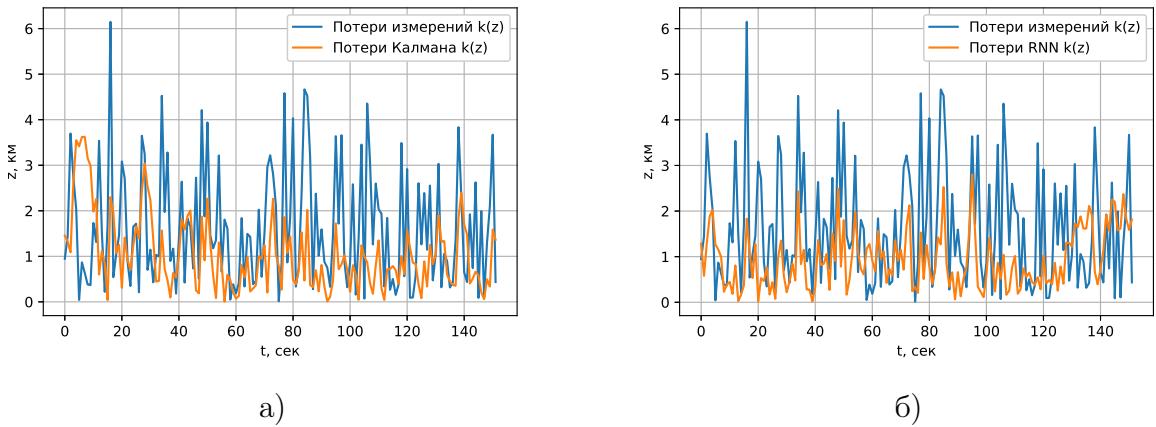


Рис. 18: Графики ошибки по оси z . а) для классического фильтра Калмана; б) для фильтра Калмана с NN.

Из графиков (16), (17), (18) видно, что ошибка фильтрации алгоритмом Калмана с нейронной сетью ниже, чем у классического. Но из-за неверных предсказаний о первой координате, неточность у фильтра с NN в начале может быть выше, чем у стандартного фильтра.

Качественно рассмотрим графики (19), (20), (21) проекций траектории на плоскости XY , XZ , YZ :

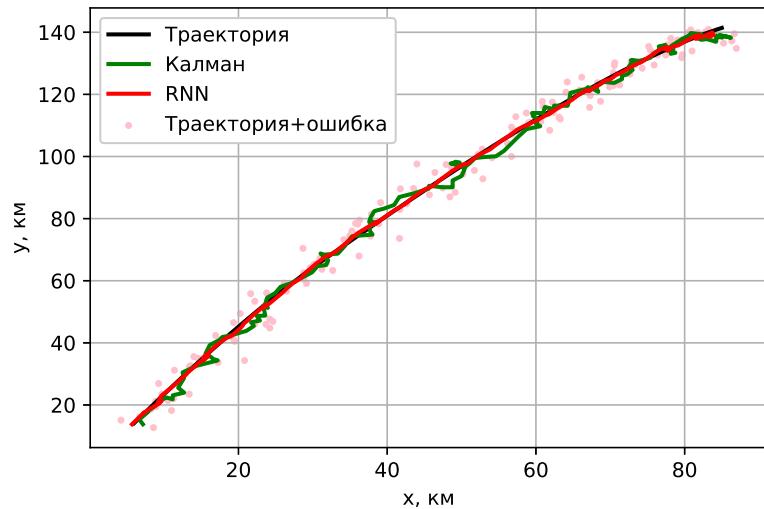


Рис. 19: Средняя точность по траекториям

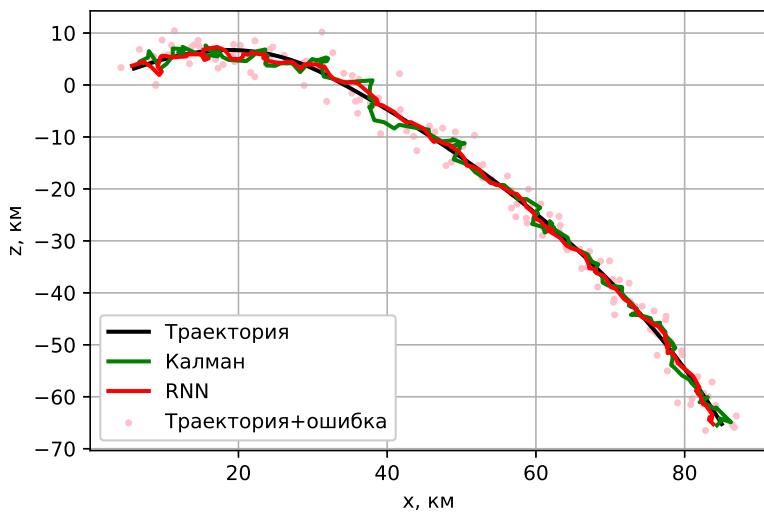


Рис. 20: Средняя точность по траекториям

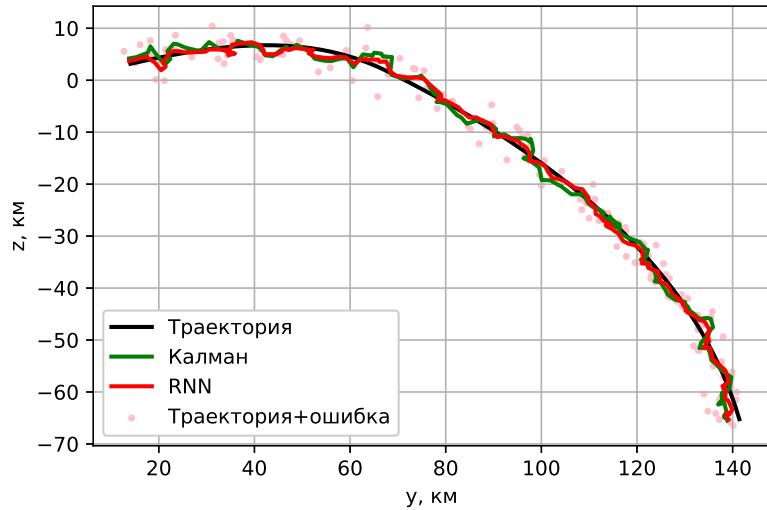


Рис. 21: Средняя точность по траекториям

Оба фильтра хорошо прибезжают сигнал к истинному, но фильтр с нейронной сетью, полагаясь на «память», лучше обрабатывает траекторию, самостоятельно подстраиваясь под спектр шума.

6.2 Фильтрация траекторий гиперзвуковых измерений рекуррентной нейронной сетью.

Смоделируем пример траектории гиперзвукового объекта (рис. 22).

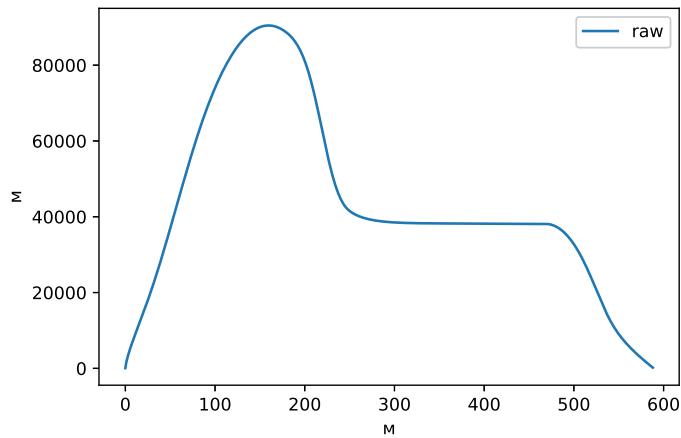


Рис. 22: Смоделированная траектория гиперзвуковой ракеты

Перед анализом набора различных траекторий изменим(улучшим) предварительную обработку данных. Новыми задачами обработки будут служить:

1. Смещение траекторий в начальную точку с нулевыми координатами;

2. Отражение траекторий в положительную плоскость;
3. Поворот координат для совмещения начальных траекторий;
4. Нормирование координат.

Данный вид обработки траекторий позволил улучшить фильтрацию данных и минимизировать ошибку отклонения, сделав её меньше чем у результатов после фильтрации классическим фильтром Калмана (см. гл. "Сравнение результатов фильтраций нейронной сетью и фильтром Калмана").

6.2.1 Обучение рекуррентной нейронной сети.

Процесс обучения и валидации аналогичен, как для баллистических объектов (см. гл. "Обучение рекуррентной нейронной сети" для баллистических траекторий). Приведём характерные для обучения графики ошибки и валидации в зависимости от эпох:

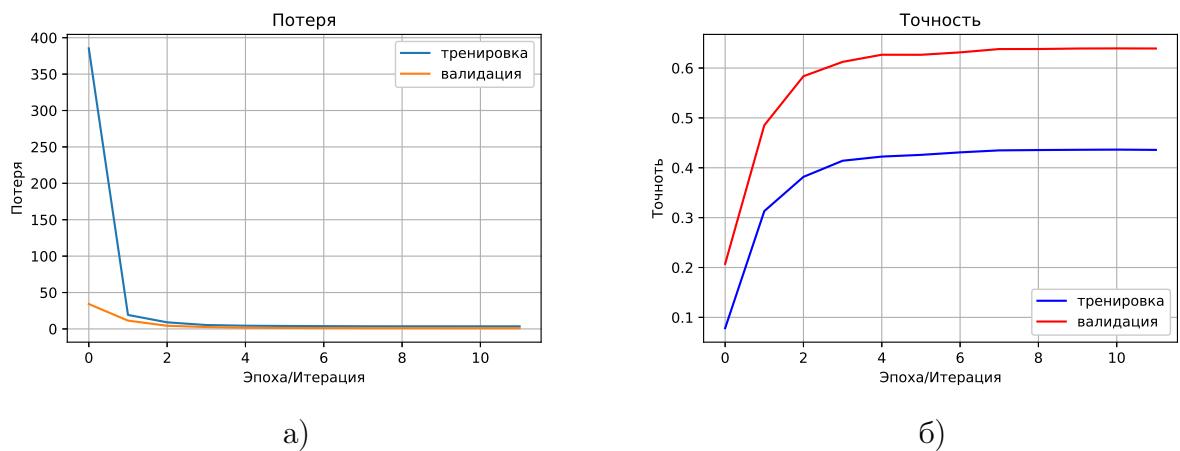


Рис. 23: Процесс обучения и валидации; а) график зависимости средней по данным MSE ошибки от эпохи; б) график зависимости средней по данным точности от эпохи.

Веса параметров фильтра подбираются методом стохастического градиентного спуска от функции потерь MSE(формула 10):

$$\text{MSE loss: } L(y_t, y_p) = (y_t - y_p)^2.$$

Потери и точность вычисляем по формулам (13), (14) из главы "Обучение рекуррентной нейронной сети" для траекторий баллистических объектов.

Вывод: Из падения графиков на MSE и росте на Accurasy видно, что сеть обучается, подтверждая необходимость и верность предварительной обработки траекторий. Точность на валидации чуть выше 63.9%, это связано с регуляризацией¹² модели и с ложностью поведения траектории.

¹²В данном случае говорится об регуляризации Dropout.

6.2.2 Проверка модели рекуррентной нейронной сети на контрольных данных.

Приведём процесс тестирования в виде графиков:

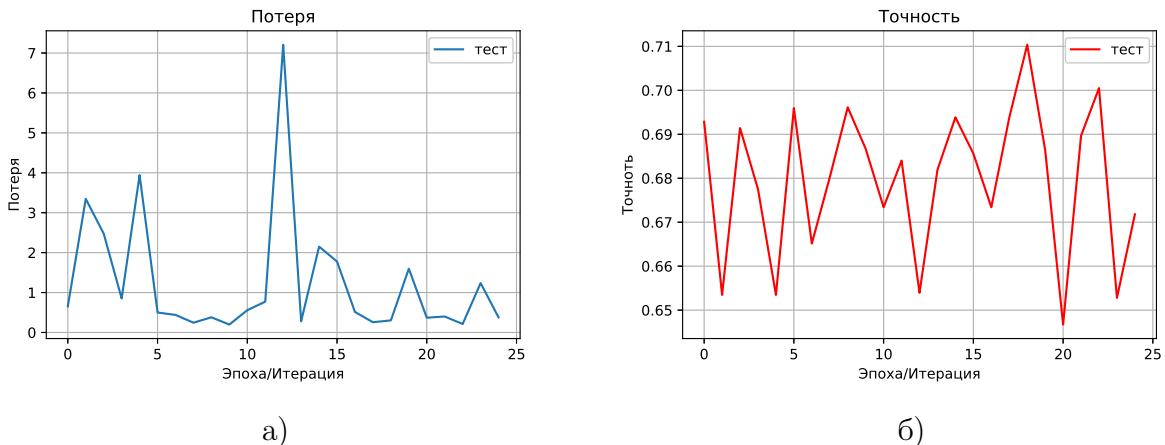


Рис. 24: Процесс тестирования. а) график зависимости средней по данным MSE ошибки от эпохи; б) график зависимости средней по данным точности от эпохи.

Ошибка в среднем 1, 2 по траекториям. А точности 5% среднем мы достигаем в 68% фильтрации.

Вывод: Сеть успешно обучилась на предварительно обработанных данных, но возможен рост характеристик точности при увеличении обучающей выборки и количества весовых параметров.

6.2.3 Сравнение результатов фильтраций нейронной сетью и фильтром Калмана.

Сравним результаты фильтраций траекторий полетов гиперзвуковых объектов, уже привычными, функциями (13), (14) (из главы "Обучение рекуррентной нейронной сети" для траекторий баллистических объектов.) обученным фильтром и уже имеющимся фильтром Калмана, представленным в главе "Математическая модель фильтра Калмана". Формулы сравнения алгоритмов:

$$loss = \sqrt{\frac{1}{S} \sum (\hat{\mathbf{x}} - \mathbf{x}_{\text{real}})^2},$$

$$accuracy = \frac{1}{S} \sum \left(\frac{|\hat{\mathbf{x}} - \mathbf{x}_{\text{real}}|}{\mathbf{x}_{\text{real}}} \leq D \right),$$

где $\hat{\mathbf{x}}$ – вектор оценочных значений фазового вектора, \mathbf{x}_{real} – действительный фазовый вектор, $D = 0,05$ – допустимая ошибка 5%, S – количество элементов в фазовом векторе.

Средние по траекториям потери RNN фильтра Калмана составляют 1, 2 км, тогда как у классического девяти-мерного фильтра они равны 1, 7 км (Рис. (25)).

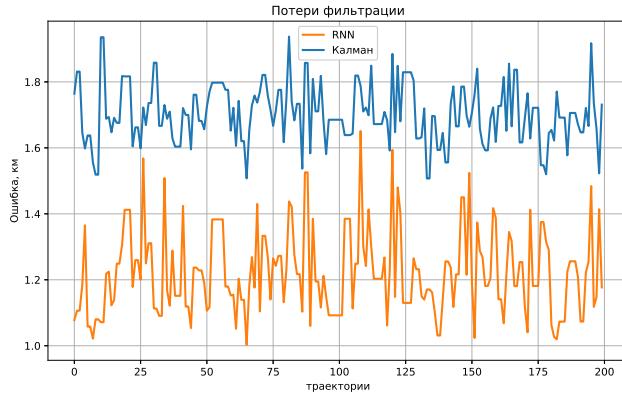


Рис. 25: Средние MSE потери по траекториям

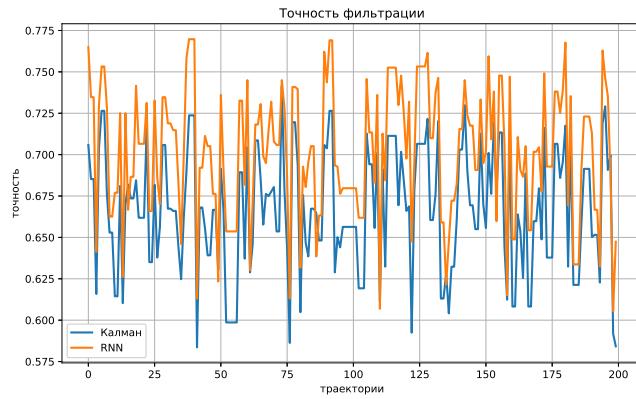
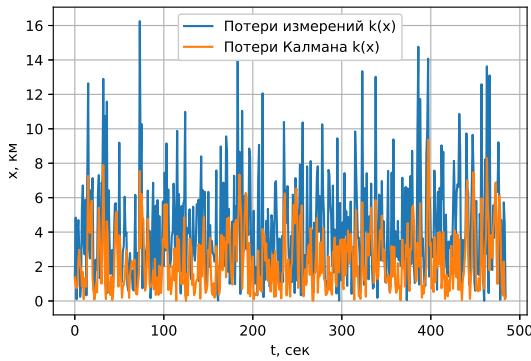


Рис. 26: Средняя точность по траекториям

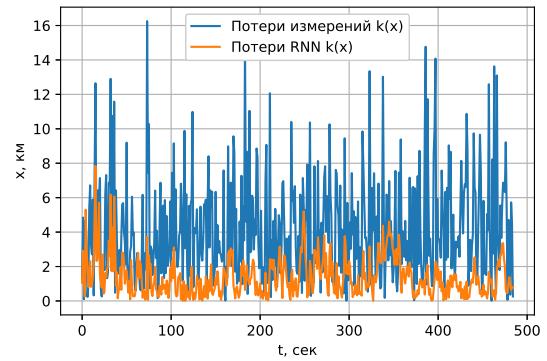
В 5%-ую точность в среднем по траектории попадают 72% результатов фильтрации RNN, и 66% для фильтра Калмана (Рис. (26)).

Для исследования характеристик фильтров рассмотрим их работу не в среднем, а на конкретной произвольной траектории. Будем выводить графики вида: |«сигнал»-«сигнал+шум»| и рядом |«фильтрованный сигнал»-«сигнал+шум»| в зависимости от времени.

Из графиков (27), (28), (29) видно, что ошибка фильтрации алгоритмом Калмана с нейронной сетью ниже, чем у классического. Но из-за неверных предсказаний о первой координате, неточность у фильтра с NN в начале может быть выше, чем у стандартного фильтра.

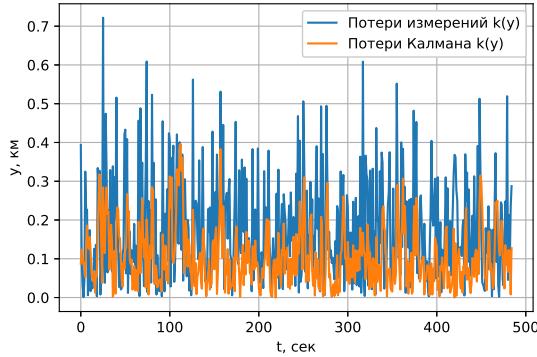


а)

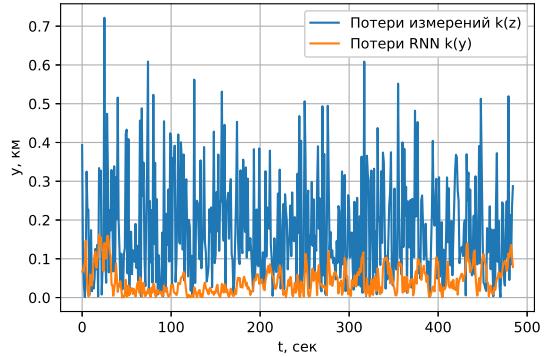


б)

Рис. 27: Графики ошибки по оси x . а) для классического фильтра Калмана; б) для фильтра Калмана с NN.

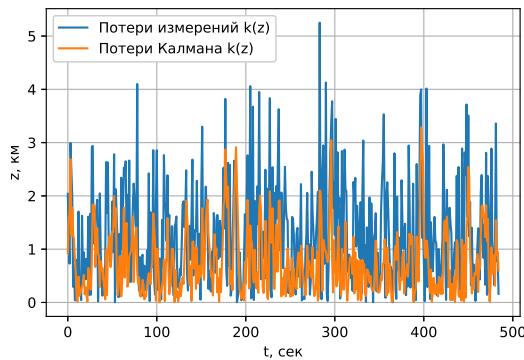


а)

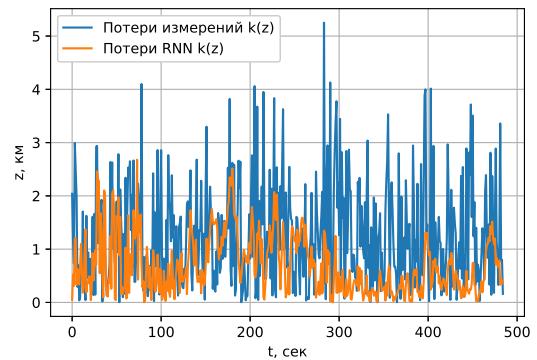


б)

Рис. 28: Графики ошибки по оси y . а) для классического фильтра Калмана; б) для фильтра Калмана с NN.



а)



б)

Рис. 29: Графики ошибки по оси z . а) для классического фильтра Калмана; б) для фильтра Калмана с NN.

Качественно рассмотрим графики (30), (31), (32) проекций траектории на плоскости XY , XZ , YZ :

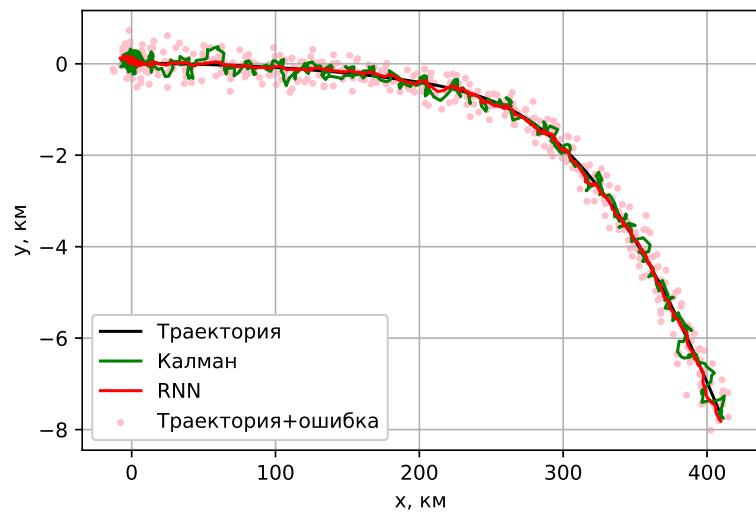


Рис. 30: Средняя точность по траекториям

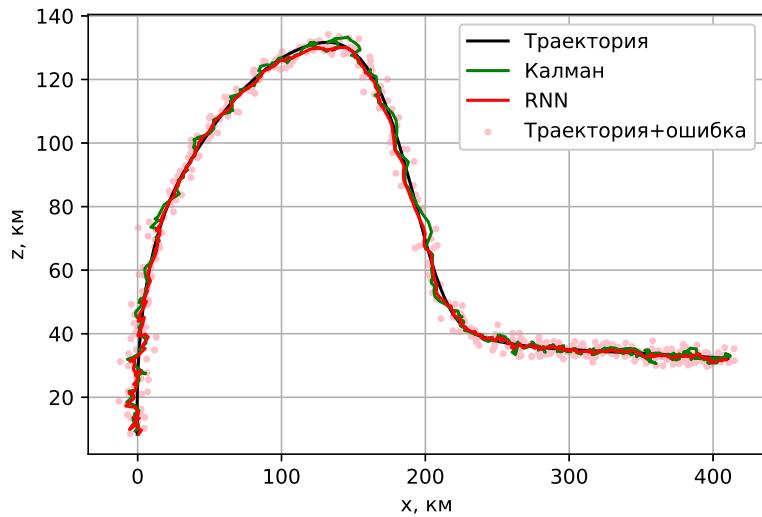


Рис. 31: Средняя точность по траекториям

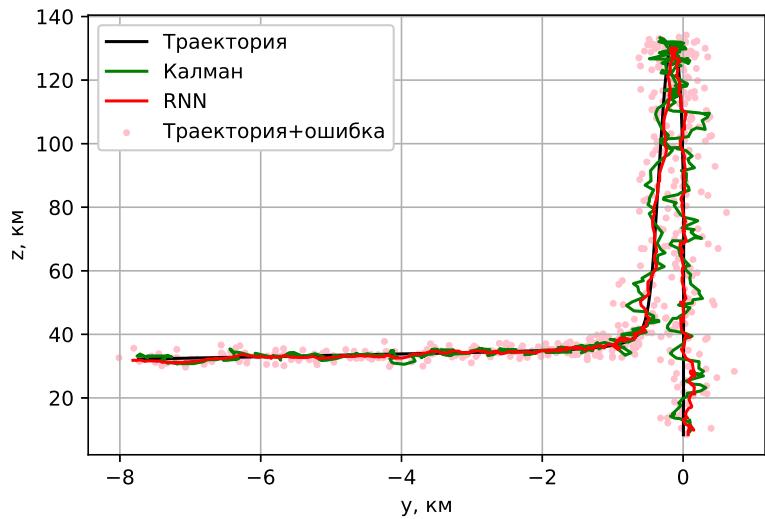


Рис. 32: Средняя точность по траекториям

Фильтр с нейронной сетью имеет существенно более низкую ошибку определения траектории движения за счет эффекта памяти при обучении.

7 Заключение.

Цель работы достигнута благодаря выполнению поставленных задач. Разработан метод снижения ошибки определения траектории и, реализующий его, алгоритм фильтрации траекторий рекуррентной нейронной сетью с уравнением экстраполяции Калмана. Проведён сравнительный анализ фильтраций нейронной сетью и рекурсивным фильтром Калмана.

- Средняя квадратичная ошибка по траекториям баллистических объектов для фильтрации нейронной сетью составляют 0,9 км, у классического рекурсивного фильтра Калмана они равны 1,2 км;
- В 5%-ую ошибку в среднем по траектории баллистических объектов попадают 81% результатов фильтрации RNN, и 73% для фильтра Калмана;
- Средние по траекториям гиперзвуковых объектов потери RNN составляют 1,2 км, у классического рекурсивного фильтра Калмана они равны 1,7 км;
- В 5%-ую ошибку в среднем по траектории гиперзвуковых объектов попадают 72% результатов фильтрации RNN, и 66% для фильтра Калмана;

Предложенный автором, метод фильтрации может найти применение у разработчиков обработки фазовых векторов. В перспективе алгоритм может иметь улучшения, связанные со снижением ошибки определения траектории и с фильтраций траекторий иных физических объектов. Исследование предполагает более глубокое изучение и сравнение других отличительных характеристик фильтров.

8 Список литературы

- [1] Книга **Roger Labbe** «Kalman and Bayesian Filters in Python» Research Institute for Advanced Study 2014г;
- [2] Статья **R. E. KALMAN** «A New Approach to Linear Filtering and Prediction Problems» 1960г;
- [3] Дипломная работа **Фонарев А. Ю.** «Машинное обучение с категориальными признаками.» МГУ имени М.В. Ломоносова 2014г;
- [4] Курс лекций **K. В. Воронцов** «Математические методы обучения по прецедентам (теория обучения машин).» www.MachineLearning.ru;
- [5] Книга **Hui Zou, Trevor Hastie** «Regression shrinkage and selection via the elastic net, with applications to microarrays» department of Statistics, Stanford University 2004г;
- [6] Книга **Spokoiny Vladimir, Dickhaus Thorsten** «Basics of Modern Mathematical Statistics» Springer Texts in Statistics 2015г;
- [7] Методическое пособие по вычислению матричных производных **Justin Johnson** «Derivatives, Backpropagation, and Vectorization» cs231n.stanford.edu 2017г;
- [8] Книга **Richard S. Sutton, Andrew G. Barto** «Reinforcement Learning: An Introduction second edition» The MIT Press Cambridge, Massachusetts 2015г;
- [9] Курс по машинному обучению МФТИ(НИУ) «Курс по машинному обучению.» 2014г;
- [10] Лекция 7 «Natural Language Processing with Deep Learning CS224N/ Ling28.» web.stanford.edu 2019г;
- [11] Документ конференции International Conference on Neural Networks; **M.W. Owen, S.C. Stubberud** «Interacting multiple model tracking using a neural extended Kalman filter.» IEEE 1999г;
- [12] Статья из рецензируемого журнала **Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, Yann N. Dauphin** «Convolutional Sequence to Sequence Learning» cornell.edu 2017г;
- [13] Статья из рецензируемого журнала **Anastasia Borovykh, Sander Bohte, Cornelis W. Oosterlee** «Conditional Time Series Forecasting with Convolutional Neural Networks » cornell.edu 2018г;

- [14] Книга **Nishant Nikhil, Brendan Tran Morris** «Convolutional Neural Network for Trajectory Prediction» Springer International Publishing 2019г;