ФГАОУВО «Московский физико-технический институт
(национальный исследовательский университет)»
Физтех-школа аэрокосмических технологий
Кафедра компьютерного моделирования

Направление подготовки: 01.03.02 Прикладные математика и информатика (бакалавриат)
Направленность (профиль) подготовки: Прикладные математика и информатика
Форма обучения: очная

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

«Прогнозирование деградации турбовентиляторных реактивных двигателей с помощью методов глубинного обучения»

(бакалаврская работа)

Студент:
Теслюк Никита Александрович

Научный руководитель:
к. ф.-м. н.
Аврутский Всеволод Игоревич

Жуковский
2021

Содержание

1 Введение

Многие авиакомпании в последние годы уже получили выгоду от использования искусственного интеллекта (AI — *artificial intelligence*) для мониторинга состояния двигателя (EHM — *engine health monitoring*) и приступили к кампаниям по оптимизации программ, призванным сдерживать расходы и повышать надежность летательных судов. Но, стоит отметить, что относительно немногие преуспели в первых попытках применить ИИ к гораздо более сложным средам обслуживания компонентов и линий, такие как двигатели. Прогностическая сила ИИ может способствовать резкому увеличению прибыльности авиакомпаний за счет устранения сбоев в работе, вызванных техническим обслуживанием, а также повышению безопасности пассажиров.

Анализируя массив исторических данных по техническому обслуживанию воздушных судов и показаний датчиков двигателей, возможна разработка системы, которая позволит прогнозировать в долгосрочной перспективе возможные дефекты по каждому отдельно взятому самолёту. Данная система способна определять вероятность возникновения разных типов дефектов в определенный период в будущем. В случае, если вероятность оказывается выше установленного порога, рекомендуется провести дополнительную проверку воздушного судна и принять должные меры по устранению неполадок.

Исходя из того, что данные, снятые с датчиков, установленных в разных частях двигателя, в большинстве своём являются временными

рядами, возникает идея применения современных нейросетевых state-of-the-art подходов глубинного обучения, используемых для работы с последовательностями, таких как рекуррентные нейронные сети и их модификации (RNN, LSTM). Также в долгосрочной перспективе для повышения точности предсказания возможно использование подходов, базирующихся на технологии Attention [1], которая широко применяется в области NLP (*natural language processing*), а также различных архитектурах типа Transformer [1].

В данной работе рассматривается применение модели LSTM (*Long Short Term Memory*). Главным преимуществом LSTM моделей над классическими RNN является то, что они способны "запоминать"долгосрочные зависимости. Эти модели очень хорошо работают с большим спектром задач и в настоящее время широко используются.

Таким образом, **основной целью** данной работы является предсказание количества оставшихся рабочих циклов двигателей до отказа, с использованием прогнозирующей модели на основе LSTM. Также в работе приведены сравнения рассматриваемой модели с классическими алгоритмами машинного обучения для решения задач регрессии и прогнозирования, таких как случайный лес, градиентный бустинг и др.

С точки зрения практического исполнения данная работа включает в себя имплементацию модели LSTM. Код, воспроизводящий эксперименты, размещен по ссылке.

2 Обзор литературы

Здесь будет обзор основной литературы, используемой в работе

1)$http://www.machinelearning.ru/wiki/images/a/a1/BayesML-2009-1.pdf$

2)$http://www.machinelearning.ru/wiki/images/e/ed/MOTP14_1.pdf$

3)$http://colah.github.io/posts/2015-08-Understanding-LSTMs/$

4)$https://www.kaggle.com/behrad3d/nasa-cmaps$

3 Задача прогнозирования

3.1 Основные понятия и определения

Задача прогнозирования исторически возникла при исследовании временных рядов и попытке предсказания их значений через какой-то промежуток времени.

В классической задаче прогнозирования обучающая выборка представляет собой набор измерений $X = \{\mathbf{x}[i]\}_{i=1}^{n}$, представляющих собой вектор веществennозначных величин $\mathbf{x}[i] = (x_1[i], ..., x_d[i])$, $\mathbf{x}[i] \in \mathbb{R}^d$ сделанных в определенные моменты времени.

Требуется постросить алгоритм (предиктор), который вернул бы точечную оценку $\{\hat{\mathbf{x}}[i]\}_{i=n+1}^{n+q}$, доверительный интервал $\{\mathbf{x}_-[i], \mathbf{x}_+[i]\}_{i=n+1}^{n+q}$ или апостериорное распределение $\mathbb{P}(\mathbf{x}[n+1], ..., \mathbf{x}[n+q] \mid \mathbf{x}[1], ..., \mathbf{x}[n])$ прогноза на заданную глубину q.

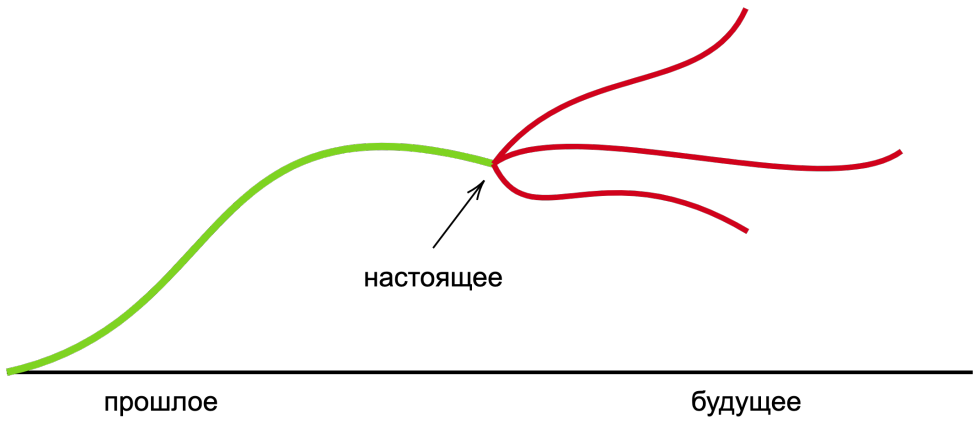


Рис. 1: Иллюстрация прогноза по времени в задачах прогнозирования

Примеры задач прогнозирования:

- Биржевое дело: прогнозирование биржевых индексов и котировок.
- Системы управления: прогноз показателей работы реактора по данным телеметрии.
- Экономика: прогноз цен на недвижимость
- Демография: прогноз изменения численности различных социальных групп в конкретном ареале
- Гидрометеорология: прогноз геомагнитной активности

3.2 Постановка задачи

В рассматриваемой задаче цель состоит в том, чтобы предсказать оставшийся полезный срок службы (RUL — *Remaining Useful Life*) каждого двигателя в наборе тестовых данных. RUL эквивалентно количеству полетов, оставшихся для двигателя после последней точки данных в тестовом наборе данных.

3.2.1 Описание эксперимента

Наборы данных состоят из нескольких временных рядов. Данные представляют собой показания датчиков в двигателях. Каждый набор данных далее делится на обучающие и тестовые выборки. Каждый временной ряд относится к определенному отдельно взятому двигателю, т. е. можно считать, что данные относятся к парку двигателей одного и того

же типа. Каждый двигатель запускается с разной степенью начального износа и производственными отклонениями, которые неизвестны пользователю. Этот износ и отклонения считаются нормальными, т. е. не считаются дефектом или неисправностью. Есть три рабочих окржунеия (в наборе данных указано, как *operational setting*), которые существенно влияют на работу двигателя. Эти настройки также включены в данные. Данные загрязнены шумом датчика.

Двигатель работает должным образом в начале каждого временного ряда и в какой-то момент времени выдает неисправность. В обучающей выборке неисправность нарастает до отказа системы. В тестовом наборе данных временной ряд заканчивается за некоторое время t до сбоя системы. Цель задачи — спрогнозировать количество оставшихся рабочих циклов до отказа в испытательной установке, то есть количество рабочих циклов после последнего текущего цикла, в которых двигатель будет продолжать работать. Также предоставлен вектор истинных значений оставшегося полезного срока службы (RUL), дабы сверить предсказанные и истинные значения и провести последующий анализ модели.

3.2.2 Описание набора данных

Данные содержат 26 столбцов, содержащих числа, разделенные пробелами. Каждая строка представляет собой массив данных, снятых в течение одного рабочего цикла, каждый столбец — это отдельная переменная.

Колонки содержат:

1. unit number — номер двигателя
2. time, in cycles — время в циклах
3. operational setting 1 — режим полёта 1
4. operational setting 2 — режим полёта 2
5. operational setting 3 — режим полёта 3
6. sensor measurement 1 — измерение датчика 1
7. sensor measurement 2 — измерение датчика 2

   ...

26. sensor measurement 26 — измерение датчика 26

3.3 Сведение к задаче глубинного обучения

Традиционные подходы к решению задач прогнозирования временных рядов включают в себя регрессионные модели прогнозирования, авторегрессионные модели прогнозирования (ARIMAX, GARCH, ARDLM), модели по выборке максимального подобия (MMSP), модели на основе генетического алгоритма (GA) и множество других. Все они так или иначе устарели или имеют множество своих особенностей и недостатков.

В последние годы развитие в области искусственного интеллекта и глубинного обучения привело к тому, что новые алгоритмы на основе нейронных сетей в разы превосходят уже существующие традиционные

подходы к решению широкого спектра задач. И одним из таких алгоритмов является модель LSTM — особый вид RNN, способный запоминать долгосрочные зависимости.

4 LSTM

4.1 Проблема долговременных зависимостей

Одним из преимуществ рекуррентных нейронных сетей (RNN) является то, что они способны сохранять сжатую историческую информацию с предыдущих слоёв сети и подавать её на вход текущему слою, что в свою очередь критически повышает точность используемой модели.

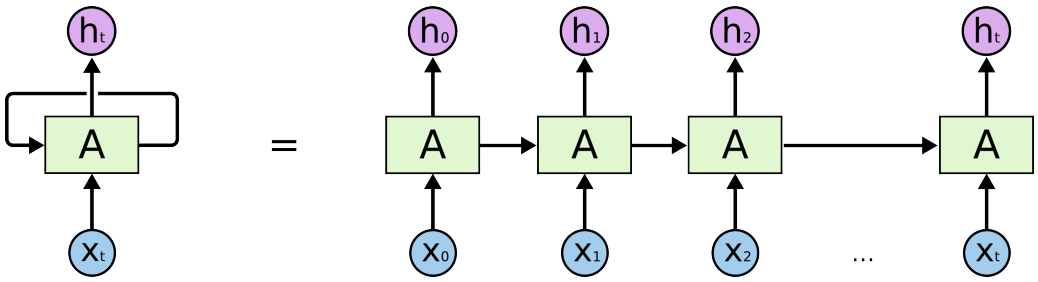


Рис. 2: Развернутая рекуррентная нейронная сеть. Взято из [1]

На приведенной выше диаграмме фрагмент нейронной сети A, на вход которой подается некоторый вектор x_t и на выходе извлекатеся вектор скрытого состояния h_t. Таким образом рекуррентную нейронную сеть можно рассматривать как несколько копий одной и той же сети, каждая из которых передает сообщение своему преемнику на следующий шаг. Это демонтрирует, что рекуррентные нейронные сети тесно связаны с последовательностями и списками и отлично подходят для работы с данными таких типов.

Иногда для выполнения какой-либо задачи нам необходима только “недавняя” информация. Например, рассмотрим языковую модель, пытающуюся предсказать следующее слово на основании предыдущих. Если

мы хотим предсказать последнее слово в предложении “облака плывут по небу”, достаточно контекста длинной в пару слов по обе стороны от интересующего нас слова; в этом случае довольно очевидно, что последним словом будет “небу”. Видно, что когда дистанция между актуальной информацией и местом, где она понадобилась, невелика, RNN могут обучиться довольно неплохо использую информацию из нескольких предыдущих слоёв.

Но бывают случаи, когда нам необходимо больше контекста. Допустим, мы хотим предсказать последнее слово в тексте “Я вырос во Франции <какой-то текст> Я бегло говорю по-французски”. Ближайший контекст предполагает, что последним словом будет называние языка, но чтобы установить, какого именно языка, нам нужен контекст, указывающий Францию из более отдаленного прошлого. Таким образом, разрыв между актуальной информацией и точкой ее применения может стать очень большим. К сожалению, по мере роста этого расстояния, RNN теряют способность связывать информацию.

В теории проблемы с обработкой долговременных зависимостей у RNN быть не должно. Человек может аккуратно подбирать параметры сети для решения задач различных типов. На практике же обучить RNN этим параметрам кажется невозможным. Эту проблему подробно исследовали Зепп Хохрайтер (Sepp Hochreiter, 1991 [2]) и Йошуа Бенджио (Yoshua Bengio) с соавторами (1994 [2]); они нашли неоспоримые причины, по которым это может быть невыполнимой задачей.

4.2 LSTM сеть

Долгая краткосрочная память (Long short-term memory; LSTM) — особая разновидность архитектуры рекуррентных нейронных сетей, способная к обучению долговременным зависимостям. Они были представлены Хохрайтером и Шмидхубером в 1997 [2] году, а затем усовершенствованы и популяризированы в работах многих других исследователей. Они хорошо справляются с целым рядом разнообразных задач и в настоящее время широко используются в области естественного языка.

LSTM разработаны специально для избежания проблемы долговременной зависимости. Запоминание длинного контекста и сохранение информации на долгие периоды времени — это их главное свойство. Любая рекуррентная нейронная сеть имеет форму цепи повторяющихся модулей нейронной сети. В обычной RNN структура одного такого модуля очень проста, например, он может представлять собой один слой с функцией активации типа сигмоида или гиперболический тангенс:

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} = 1 - S(-x),$$

$$f(x) = \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x}-1}{e^{2x}+1}$$

В свою очередь LSTM-структура также напоминает цепь, но внутренние модули выглядят иным образом. Вместо одного слоя нейронной сети они содержат целых четыре, и эти слои взаимодействуют особенным

образом.

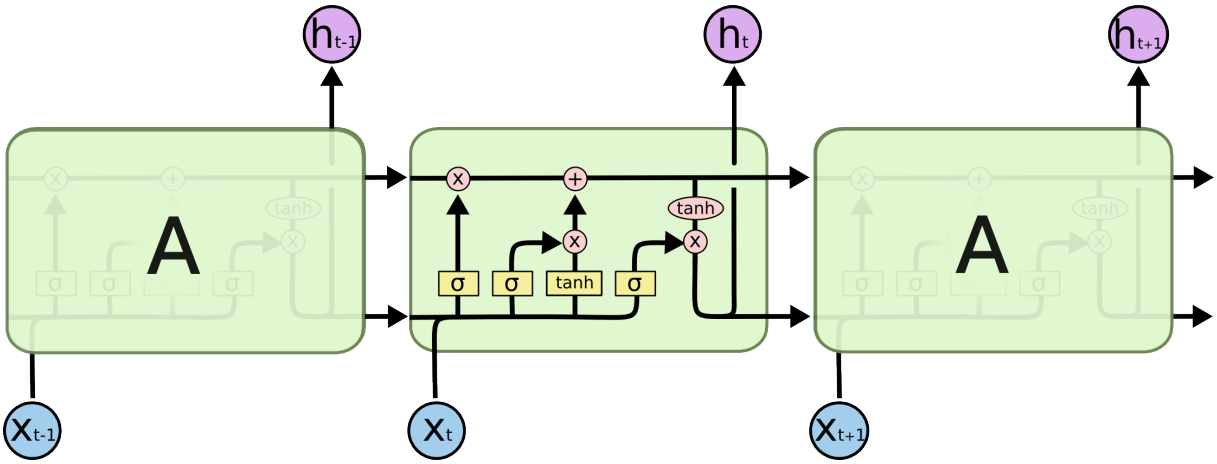


Рис. 3: Модуль LSTM содержит четыре слоя. Взято из [1]

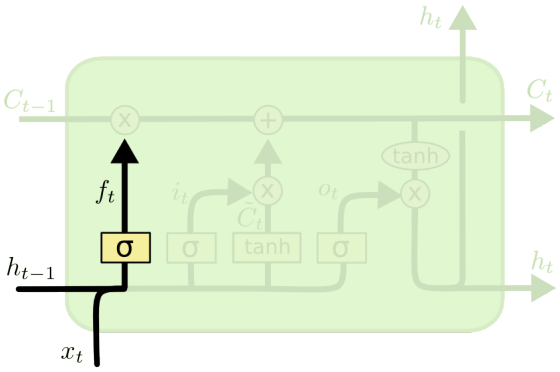
На схеме 3 каждая стрелка переносит вектор от выхода одного модуля и подаёт его на вход следующего. Красными кружочками обозначены поточечные операции, такие, как сложение или перемножение векторов. Желтые прямоугольники соответствуют слоям нейронной сети. Сходящиеся стрелки означают объединение, а разветвляющиеся стрелки отвечают за копирование данных и их передачу в другие компоненты сети.

Ключевой концепт LSTM — состояние ячейки (*cell state*) — горизонтальная линия, проходящая по верхней части схемы. Состояние ячейки проходит напрямую через всю цепочку, участвуя лишь в нескольких линейных преобразованиях. Информация может легко течь по ней, не подвергаясь изменениям. Также LSTM может удалять информацию из состояния ячейки; этот процесс регулируется так называемыми фильтрами (*gates*). Фильтры позволяют пропускать информацию на основании некоторых условий. Они состоят из слоя сигмоидальной нейронной сети и операции поточечного умножения. Сигмоидальный слой возвращает чис-

ла от нуля до единицы, которые обозначают, какую долю каждого блока информации следует пропустить дальше по сети. Ноль в данном случае означает “не пропускать ничего”, единица — “пропустить все”. LSTM имеет три таких фильтра, позволяющих защищать и контролировать состояние ячейки.

4.3 Шаги работы

1. Первый шаг в LSTM — определить, какую информацию можно выбросить из состояния ячейки. Это решение принимает слой сигмоиды, называемый “слоем фильтра забывания” (forget gate layer). Он смотрит на h_{t-1} и x_t и возвращает число от 0 до 1 для каждого числа из состояния ячейки C_{t-1}. Единица означает “полностью сохранить”, а ноль – “полностью выбросить”.

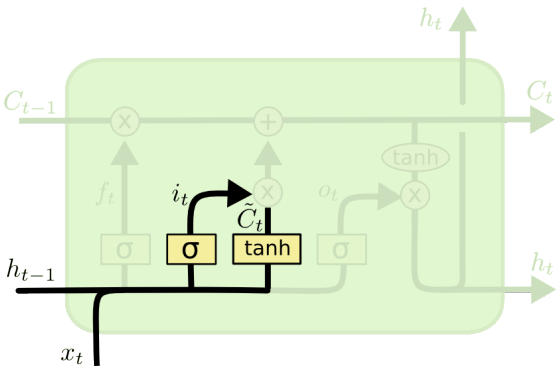


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Рис. 4: Первый шаг. Взято из [1]

2. Второй шаг — решить, какого рода новая информация будет храниться в состоянии ячейки. Сначала “слой входного фильтра” (*input layer gate*) определяет, какие значения следует обновить. За-

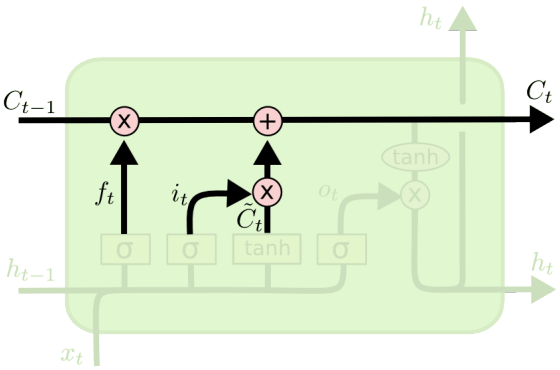
тем tanh-слой строит вектор новых значений-кандидатов $\widetilde{C}_t$, которые можно добавить в состояние ячейки.



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

Рис. 5: Второй шаг. Взято из [1]

3. Третьим шагом требуется заменить старое состояние ячейки C_{t-1} на новое C_t. Для этого нужно умножить старое состояние на f_t, тем самым забывая ненужную информацию. Затем прибавить $i_t * \widetilde{C}_t$, чтобы получить информацию о том насколько нужно обновить каждое из значений нового состояния.

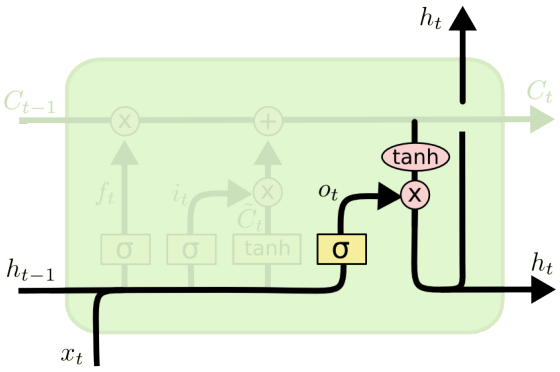


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Рис. 6: Третий шаг. Взято из [1]

4. На последнем шаге нужно решить, какая информация должна быть на выходе. Выходные данные будут основаны на текущем

состоянии ячейки. Сначала требуется применить слой сигмоиды, который определяет, какую информацию из состояния ячейки мы будем выводить. Затем применяется tanh-слой, чтобы получить на выходе значения из диапазона от -1 до 1, которые перемножаются с выходными значениями слоя сигмоиды, что позволяет выводить только требуемую информацию.



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

Рис. 7: Четвёртый шаг. Взято из [1]

5 Вычислительный эксперимент

5.1 Предобработка данных

Для экспериментов был выбран набор данных с сайта NASA за авторством A. Saxena, K. Goebel, D. Simon, and N. Eklund (2008) [3], содержащий:

- ∼20,000 строк в тренировочной выборке, 100 уникальных id двигателей;
- ∼13,000 строк в тестовой выборке, 100 уникальных id двигателей;
- 100 строк — ground truth метки

Из данных были удалены скореллированные признаки ($> 80\%$ корелляции), как из тренировочной выборки, так и из тестовой выборки. Далее, оба набора данных были очищены от шума, пробелы в колонках были заменены на усреднённые показания по конкретно рассматриваемому признаку. Также была произведена нормировка признаков с помощью MinMaxScaler:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

5.2 Обучение модели

В качестве модели используется глубокуя LSTM сеть. На первом слое — LSTM со 100 единицами (*units — размерность выходного простран-*

ства), за которым следует еще один слой LSTM с 50 единицами. Dropout, равный 0.2 применяется после каждого слоя LSTM для улучшения точности модели, а также во избежание переобучения. Последний слой — это полносвязная сеть с активацией типа сигмоида и одномерным выходом. Количество тренируемых параметров ∼80,000. Архитектура нейронной сети представлена на схеме 8:

Model: "sequential_1"

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 50, 100)	50400
dropout_1 (Dropout)	(None, 50, 100)	0
lstm_2 (LSTM)	(None, 50)	30200
dropout_2 (Dropout)	(None, 50)	0
dense_1 (Dense)	(None, 1)	51
activation_1 (Activation)	(None, 1)	0

Total params: 80,651
Trainable params: 80,651

Рис. 8: Архитектура нейронной сети.

Обучение производится на 80% тренировочной выборки. Остальные 20% данных в тренировочном датасете отложены для валидации модели. Дополнительно, используется механизм Early Stopping — остановка обучения нейронной сети при выходе ошибки на плато либо при её систематическом увеличении для сохранения оптимальных параметров. Сеть обучается на протяжении 100 эпох (*epoch* — полный цикл обучения). Размер батча (*batch — кусок данных для обучения на конкретном*

шаге) равен 200. В качестве метрик используются MAE (*mean absolute error*) и коэффициент детерминации R^2. Все описанные параметры являются оптимальными и были получены эксперементальным путём в ходе процедуры Grid Search.

Ниже приведён алгоритм обучения модели:

Алгоритм 1 Процедура обучения

1: Разбить тренировочную выборку на батчи $X_{batch} = \{\mathbf{x}[i]\}_{i=1}^{n}$ по 200 элементов
2: **for** t $= 1, \ldots num\_epochs$ **do**
3: Подать на вход последовательность из батча в виде вектора
4: Вычислить предсказание на текущем шаге
5: Сделать шаг градиентной оптимизации
6: Перейти на следующую итерацию
7: **end for**

Выход: усредненные значения остаточного числа циклов работы двигателя за n батчей

5.3 Валидация модели

Так как рассматриваемые данные представляют собой временные ряды, то они упорядочены относительно неслучайных моментов времени, а следовательно, в отличие от случайных выборок, могут содержать в себе дополнительную важную для обучения информацию, которую нельзя терять. Поэтому стандартные техники валидации в этом случае являются непригодными для использования. Таким образом, в процессе валидации пришлось использовать более специфический способ для оптимизации параметров, так называемый *cross-validation on a rolling basis* или кросс-валидация на скользящем окне.

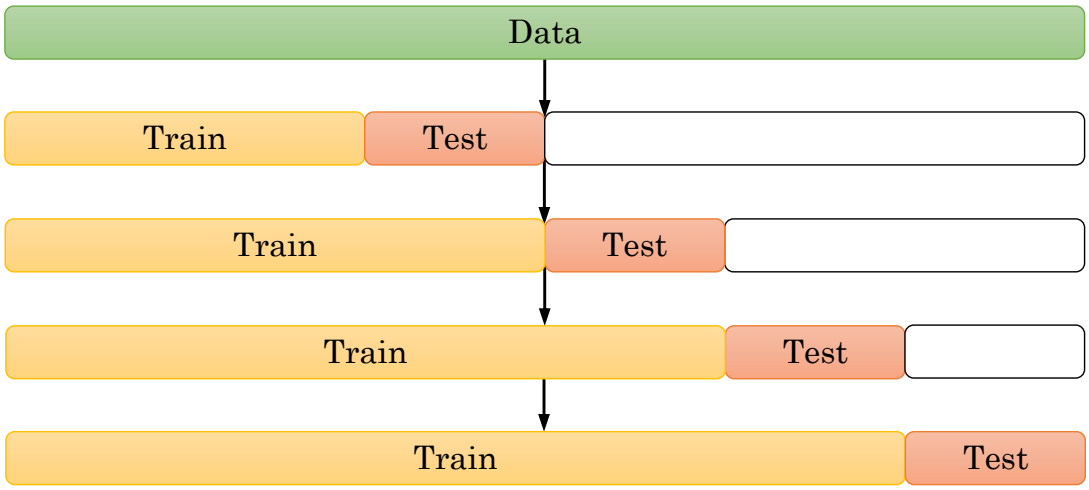


Рис. 9: Схема валидации.

Суть такого подхода состоит в том, что сначала модель обучается на небольшом отрезке временного ряда, от начала до некоторого момента времени t, затем делается прогноз на $t + n$ шагов вперед и считается ошибка. Далее обучающая выборка расширяется до значения $t+n$ и прогнозируется с $t+n$ до $t+2n$. Так продолжает двигаться валидационный отрезок ряда до тех пор, пока не упирается в последнее доступное наблюдение. В итоге получится столько фолдов — групп данных, сколько n уместится в промежуток между изначальным обучающим отрезком и всей длиной рассматриваемого ряда.

5.4 Проверка на тестовых данных

5.5 Результаты

В ходе обучения критерии качества измерялись двумя способами. Раз в 10000 шагов измерялось качество на всей тестовой выборкее, раз в 100

шагов — на одном случайно выбранном изначально пользователе.

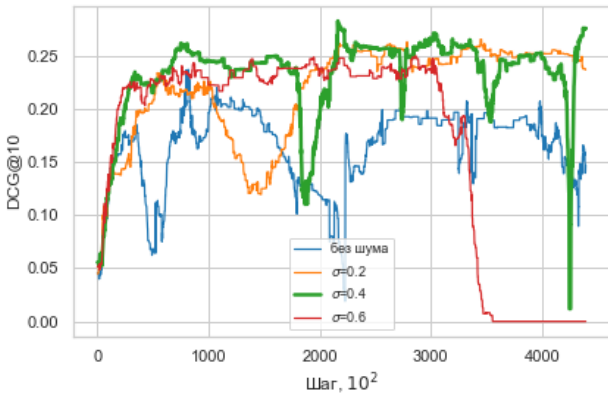


Рис. 10: Кривая обучения DCG@10 для одного пользователя

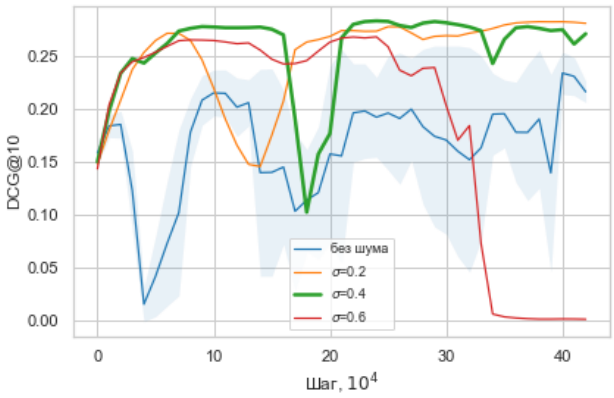


Рис. 11: Кривая обучения DCG@10 для всех пользователей

Синим затемнением на графике отмечено стандартное отклонение по трём запускам модели без шума.

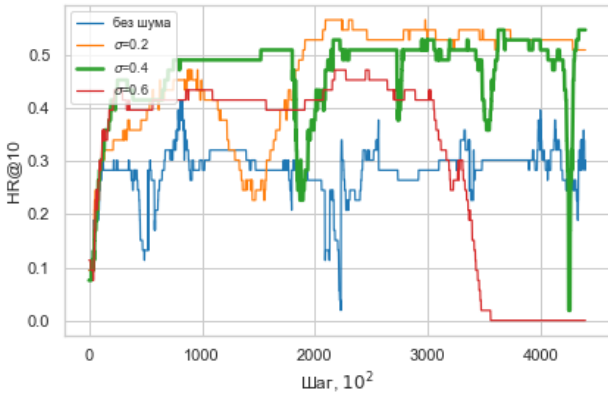


Рис. 12: Кривая обучения HR@10 для одного пользователя

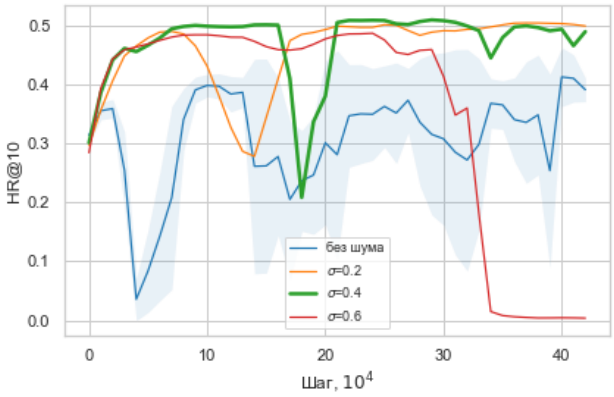


Рис. 13: Кривая обучения HR@10 для всех пользователей

Для итогового вычисления качества модели использовались наилучшие веса из истории оценивания по всем пользователям. Результаты представлены в таблице 1

Модель	DCG@10	HR@10
$\sigma = 0.6$	0.268	0.487
$\boldsymbol{\sigma = 0.4}$	**0.282**	**0.509**
$\sigma = 0.2$	**0.282**	0.504
без шума	0.254	0.454
Случайные рекомендации	~ 0.05	~ 0.1

Таблица 1: Сравнение разных вариантов шума

Видно, что использование шума повышает как итоговые критерии качества, так и их промежуточные значения в процессе обучения.

6 Заключение

6.1 Итоги работы

В рамках проведенного исследования была достигнута поставленная цель и решены сформулированные в начале исследования задачи. На защиту выносятся следующие результаты:

1. Разработана модель ранжирования рекомендаций на основе алгоритма обучения с подкреплением актор-критик с использованием стохастических процессов Орнштейна — Уленбека

2. Показано, что оптимизация дисперсии процессов Орнштейна — Уленбека улучшает качество рекомендаций по критериям DCG и HR.

3. Показано, что детерминированные предсказания затрудняют исследование среды агентом.

6.2 Дальнейшие исследования

Рассматриваются следующие возможные варианты развития данной работы:

1. Изучить подходы к построению состояний среды.

2. Сравнить рассмотренный в данной работе метод с другими упомянутыми перспективными алгоритмами, такими как trulyPPO [4].

3. Исследовать более сложные постановки задач рекомендательного моделирования, включая многошаговые рекомендательные сценарии, где на каждой итерации происходит переформулировка или уточнение запроса. Такая постановка задачи близка к разведочному поиску. Обычно разведочный поиск включает в себя несколько итераций поисковых запросов, а также используется в случаях, когда пользователь не имеет четкого запроса или представления о требуемом результате поиска. Цель такого поиска — не только найти информацию, точно соответствующую запросу, но и осознать, изучить новую тему. Обучение с подкреплением — подходящий метод для решения такой задачи. Таким образом, данное исследование может быть продолжено не только в рамках рекомендательного моделирования, но и в рамках алгоритмов поиска текстовых документов.

Также в дальнейшем планируется перейти на использование фреймворка Catalyst (включен в Pytorch Ecosystem) [**?**].

Список литературы

[1] Deep Reinforcement Learning based Recommendation with Explicit User-Item Interactions Modeling / Feng Liu, Ruiming Tang, Xutao Li et al. // ArXiv. — 2018. — Vol. abs/1810.12027.

[2] Sutton Richard S., Barto Andrew G. Reinforcement Learning: An Introduction. — Cambridge, MA, USA : A Bradford Book, 2018. — ISBN: 0262039249.

[3] A. Saxena K. Goebel D. Simon, Eklund N. Damage propagation modeling for aircraft engine run-to-failure simulation // 2008 International Conference on Prognostics and Health Management. — 2008. — Dec. — Vol. 5, no. 4. — Access mode: https://doi.org/10.1145/2827872.

[4] Wang Yuhui, He Hao, Wen Chao, Tan Xiaoyang. Truly Proximal Policy Optimization. — 2019. — 1903.07940.

[5] A Contextual-Bandit Approach to Personalized News Article Recommendation / Lihong Li, Wei Chu, John Langford, Robert E. Schapire // Proceedings of the 19th International Conference on World Wide Web. — WWW '10. — New York, NY, USA : Association for Computing Machinery, 2010. — P. 661–670. — Access mode: https://doi.org/10.1145/1772690.1772758.

[6] Deep reinforcement learning for page-wise recommendations / Xiangyu Zhao, Long Xia, Liang Zhang et al. // Proceedings of the 12th ACM Conference on Recommender Systems - RecSys '18. — ACM Press, 2018. — Access mode: https://doi.org/10.1145/3240323.3240374.

[7] Deep Reinforcement Learning for List-wise Recommendations / Xiangyu Zhao, Liang Zhang, Zhuoye Ding et al. // ArXiv. — 2018. — Vol. abs/1801.00209.

[8] Koren Yehuda, Bell Robert, Volinsky Chris. Matrix Factorization Techniques for Recommender Systems // Computer. — 2009. — Aug. — Vol. 42, no. 8. — P. 30–37. — Access mode: https://doi.org/10.1109/MC.2009.263.

[9] Google News Personalization: Scalable Online Collaborative Filtering / Abhinandan S. Das, Mayur Datar, Ashutosh Garg, Shyam Rajaram // Proceedings of the 16th International Conference on World Wide Web. — WWW '07. — New York, NY, USA : Association for Computing Machinery, 2007. — P. 271–280. — Access mode: https://doi.org/10.1145/1242572.1242610.

[10] Philip Simon, Shola Peter, Abari Ovye. Application of Content-Based Approach in Research Paper Recommendation System for a Digital Library // International Journal of Advanced Computer Science and Applications. — 2014. — 10. — Vol. 5.

[11] Kompan Michal, Bieliková Mária. Content-Based News Recommendation // E-Commerce and Web Technologies / Ed. by Francesco Buccafurri, Giovanni Semeraro. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2010. — P. 61–72.

[12] Shani Guy, Brafman Ronen I., Heckerman David. An MDP-Based Recommender System // Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence. — UAI'02. — San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2002. — P. 453–460.

[13] Schulman John, Wolski Filip, Dhariwal Prafulla et al. Proximal Policy Optimization Algorithms. — 2017. — 1707.06347.

[14] Deterministic Policy Gradient Algorithms / David Silver, Guy Lever, Nicolas Heess et al. // Proceedings of the 31st International Conference on Machine Learning / Ed. by Eric P. Xing, Tony Jebara. — Vol. 32 of Proceedings of Machine Learning Research. — Bejing, China : PMLR, 2014. — 22–24 Jun. — P. 387–395. — Access mode: http://proceedings.mlr.press/v32/silver14.html.

[15] Continuous control with deep reinforcement learning. / Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel et al. // ICLR / Ed. by Yoshua Bengio, Yann LeCun. — 2016.

[16] Fujimoto Scott, van Hoof Herke, Meger David. Addressing Function Approximation Error in Actor-Critic Methods // Proceedings of the 35th International Conference on Machine Learning, ICML 2018,

Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 / Ed. by Jennifer G. Dy, Andreas Krause. — Vol. 80 of Proceedings of Machine Learning Research. — PMLR, 2018. — P. 1582–1591. — Access mode: http://proceedings.mlr.press/v80/fujimoto18a.html.

[17] Uhlenbeck G. E., Ornstein L. S. On the theory of the Brownian motion // Phys. Rev. — 1930. — Vol. 36, no. 3. — P. 823–841.

[18] Kingma Diederik P, Welling Max. Auto-Encoding Variational Bayes. — 2013. — 1312.6114.

[19] Policy Gradient Methods for Reinforcement Learning with Function Approximation / Richard S. Sutton, David McAllester, Satinder Singh, Yishay Mansour // Proceedings of the 12th International Conference on Neural Information Processing Systems. — NIPS'99. — Cambridge, MA, USA : MIT Press, 1999. — P. 1057–1063.

[20] DRN: A Deep Reinforcement Learning Framework for News Recommendation / Guanjie Zheng, Fuzheng Zhang, Zihan Zheng et al. // Proceedings of the 2018 World Wide Web Conference. — 2018.

[21] End-to-End Deep Reinforcement Learning Based Recommendation with Supervised Embedding / Feng Liu, Huifeng Guo, Xutao Li et al. // Proceedings of the 13th International Conference on Web Search and Data Mining. — WSDM '20. — New York, NY, USA : Association for Computing Machinery, 2020. — P. 384–392. — Access mode: https://doi.org/10.1145/3336191.3371858.

[22] Dulac-Arnold Gabriel, Evans Richard, van Hasselt Hado et al. Deep Reinforcement Learning in Large Discrete Action Spaces. — 2015. — 1512.07679.