ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

УДК 519.6

Отчет об исследовательском проекте на тему:

Компьютерный анализ биологической модели адаптивной динамики

Выполнила:

студентка группы БПМИ219
Василева Анна Ярославовна _____ _____
(подпись) (дата)

Принял руководитель проекта:

Никитин Алексей Антонович
Доцент кафедры общей математики ВМК МГУ,
Кандидат физико-математических наук

_____ _____
(подпись) (дата)

Москва 2023

Содержание

Аннотация

В данной работе изучаются двухмерные одновидовые симуляции модели Дикмана-Лоу. Цель данной работы - ускорить построение графиков и проводить вычисления лишь пока численность популяции не выйдет на плато. Кроме того, нужно научиться предсказывать численность популяции на плато для двумерных симуляций по входным параметрам. Для этого вручную собран датасет, основанный на большом количестве проведенных симуляций. Для ускорения получения данных используется переделанный под двумерные симуляций алгоритм поиска точки выхода на плато.

Ключевые слова

Модель Дикмана-Лоу, двумерные симуляции, сглаживания, выход на плато, машинное обучение

1 Введение

При изучении биологических сообществ с большой продолжительностью жизни особей применяются методы математического моделирования. В работе рассматривается модель Ульфа Дикмана и Ричарда Лоу. Ее отличие от предшествующих моделей состоит в том, что она учитывает структуру сообщества в пространстве, в то время как других моделях считается, что популяция распределена по ареалу обитания равномерно.

Для изучения динамики численности популяции и других важных характеристик, например, расположения особей в пространстве, в рамках модели Дикмана-Лоу используются компьютерные симуляции, ранее реализованные научной группой. Цель моей работы состоит в оптимизации времени расчета двухмерных симуляций. Для этого к предварительно сглаженным данным о численности популяции применен метод сравнения дисперсий соседних срезов. На основании полученных результатов сформулированы критерии выхода на плато и обновлен алгоритм поиска момента выхода на плато, написанный до этого лишь для одномерных симуляций. Далее предпринята попытка применить модели машинного обучения для предсказания численности популяции на плато по заданным параметрам симуляции. Для этого вручную создан датасет из примерно 16000 примеров, на нем обучено несколько классических моделей: линейная регрессия, случайный лес, бустинг; а также подобрана нейронная сеть с оптимальной структурой. Минимальная ошибка RMSE достигнута при обучении нейронной сети и составила 235.

2 Предметная область

2.1 Модель саморазвивающихся биологических сообществ

В работе исследуется модель Дикмана-Лоу, детализированная в статьях [2], [1]. Данная модель описывает популяцию неподвижных организмов, обитающих в замкнутой области $A \subseteq R^n$, $n \in [1, 2, 3]$. В моей работе далее $A \subseteq R^2$. Все особи - материальные точки, различающиеся только положением в пространстве. Время непрерывно, и в каждый момент с каждым индивидом случается одно из двух событий: рождение либо смерть.

Рождаемость задается параметром b - плодовитостью вида и ядром рождаемости - радиально симметричной функцией, определяющей вероятность возникновения новой особи на фиксированном расстоянии от родителя. Смертность подразделяется на естественную d и смертность от конкуренции между особями dd. Вероятность смерти от конкуренции задается ядром конкуренции, учитывающим помимо dd расстояние между индивидами.

2.2 Симуляции

Алгоритм проведения симуляций написан на C++ и R. Он работает для всех случаев: одномерного, двухмерного и трехмерного пространств. Часть, написанная на R, отвечает за запуск симуляции, а часть на C++ используется для быстрого расчета событий во время каждой эпохи.

Область A задается как n-мерный куб с периодическими границами. Таким образом, если особь рождается за пределами границы, она появляется с противоположной стороны. В качестве единиц времени выступают эпохи: за каждую эпоху для каждого из индивидов происходит событие рождения либо смерти. Так как эти события - пуассоновские потоки с фиксированными рождаемостью и смертностью, можно генерировать события по очереди для каждого индивида и дальше пользоваться аддитивностью потоков. [4]

Список параметров симуляции:

Таблица 2.1: Параметры симуляции

Параметр	Значение
b	плодовитость вида
d	естественная смертность
sd_b	стандартное отклонение ядра рождаемости
sd_d	стандартное отклонение ядра смертности
dd	смертность от конкуренции
area_length_x	длина по оси x зоны обитания
area_length_y	длина по оси y зоны обитания
initial_pop	начальная популяция
epochs_count	количество эпох

3 Подготовка к созданию датасета

3.1 Настройка двухмерных симуляций

Для начала генерации симуляций необходимо было настроить ядра рождения $m(e)$ и смерти $w(e)$, так как пример запуска симуляции для двухмерного случая отсутствовал.
Они обладают следующими свойствами:

$$\forall e : w(e) \geq 0 : \int_R^n w(e)de = 1,\ \lim_{x\to\infty} w(x) = 0, \forall x, y \in R^n : ||x|| = ||y|| \implies ||w(x)|| = ||w(y)|| \tag{1}$$

$$\forall e : m(e) \geq 0 : \int_R^n m(e)de = 1,\ \lim_{x\to\infty} m(x) = 0, \forall x, y \in R^n : ||x|| = ||y|| \implies ||m(x)|| = ||m(y)|| \tag{2}$$

Во всех экспериментах ядра рождения и смерти были распределены с нулевым математическим ожиданием и дисперсией sd_b и sd_d соответственно. В двумерном случае ядро рождения было задано с помощью квантилей распределения Рейлея, а ядро смерти вычислялось для радиуса r в сетке $[0, 3sd_d]$ по формуле

$$\frac{1}{2\pi sd_d^2} \exp^{\frac{-1}{2sd_d^2} r^2} \tag{3}$$

После задания ядер код симуляций был протестирован на примере 3.1

Таблица 3.1: Параметры тестовой симуляции

Параметр	Значение
b	1
d	0
sd_b	1
sd_d	1
dd	0.01
area_length_x	10
area_length_y	10
initial_pop	1
epochs_count	1000

Выход на плато произошел при численности популяции 10000, как и ожидалось.

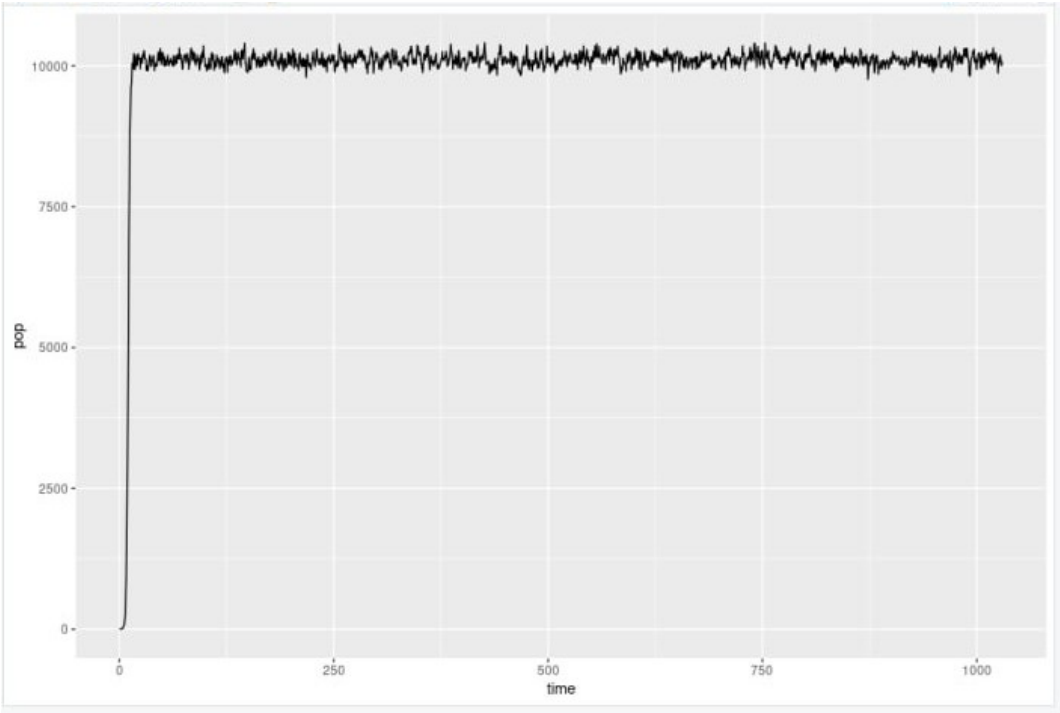

Рис. 3.1: График численности тестовой симуляции

3.2 Сглаживания результатов симуляций

Так как симуляции стохастичны, они могут колебаться около тренда. Для удаления лишней информации и выявления общей тенденции используются сглаживания. Было испытано три вида сглаживания: простое, простое экспоненциальное и двойное экспоненциальное. Код для построения сглаживаний и графиков выложен в **Приложении А**

3.2.1 Простое скользящее среднее

Для начала было использовано простое скользящее среднее: функция, сопоставляющая каждому значению численности популяции среднее численности популяции, рассчитанное по последним k точкам, где k называется размером окна. Из-за того, что для расчета необходимы k предыдущих значений, первые k значений не сглаживаются.
Таким образом,

$$pop_t = \frac{1}{k}\sum_{i=0}^{k-1} pop_{t-i}, \forall t \in [k+1, epochs] \tag{4}$$

Для сглаживания с помощью простого скользящего среднего использовалось несколько вариантов k: $k = 20, k = 50, k = 100$. Для небольших значений k, например, $k = 20$, график остается сильно зашумленным (Рисунок 3.2).

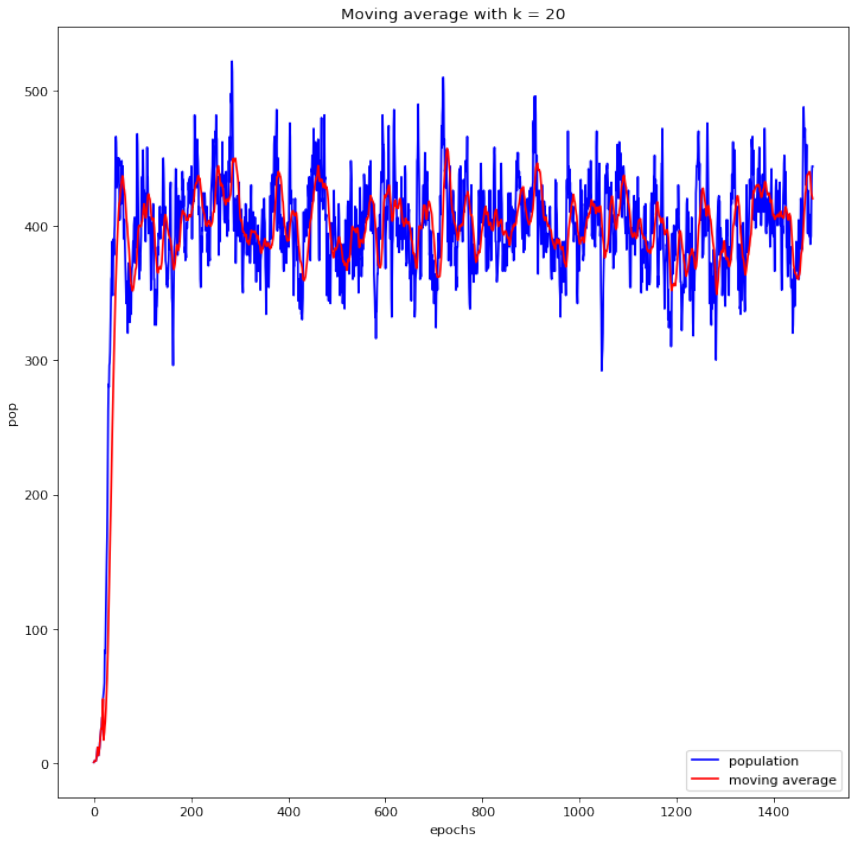

Рис. 3.2: Применение простого скользящего среднего с размером окна 20

При росте k шум начинает пропадать (Рисунок 3.3), но возникает проблема с неправильным нахождением значений для точек, строящихся по первым окнам. Ее можно устранить, например, добавив больший вес последним точкам, входящим в первое окно.

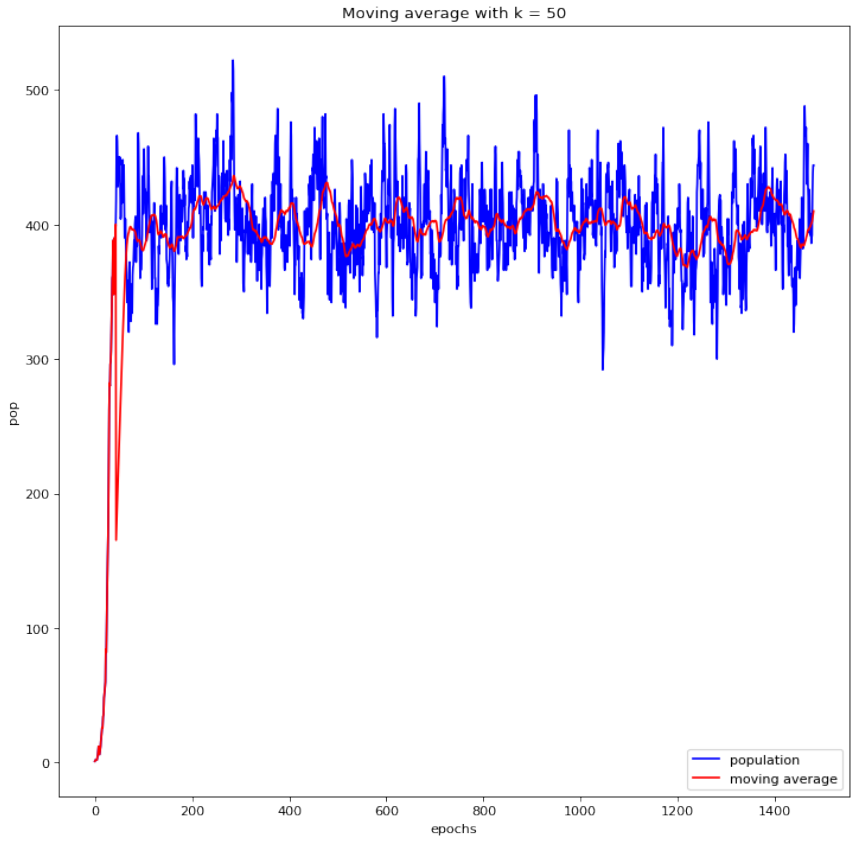

Рис. 3.3: Применение простого скользящего среднего с размером окна 50

Основной трудностью применения простого сглаживания остается сильная зависимость полученного сглаживания от k (Рисунок 3.4), так как при проведении симуляций в режиме реального времени трудно предсказать, как долго они будут длиться: некоторые симуляции выходят на плато за 50-60 эпох, а некоторым требуется несколько тысяч эпох. В связи с тем, что сложно подобрать универсальное правило для вычисления размера окна, было принято решение не использовать этот метод.

3.2.2 Простое экспоненциальное сглаживание

Далее пробовалось простое экспоненциальное сглаживание, задающееся формулой

$$\hat{p}_{t+1} = \alpha p_{t+1} + (1 - \alpha)\hat{p}_t, \tag{5}$$

где $\alpha \in (0, 1)$ - параметр, отвечающий за силу сглаживания: чем выше α, тем больше вес у последних наблюдений; p_t - исходное значение численности популяции в момент времени t,

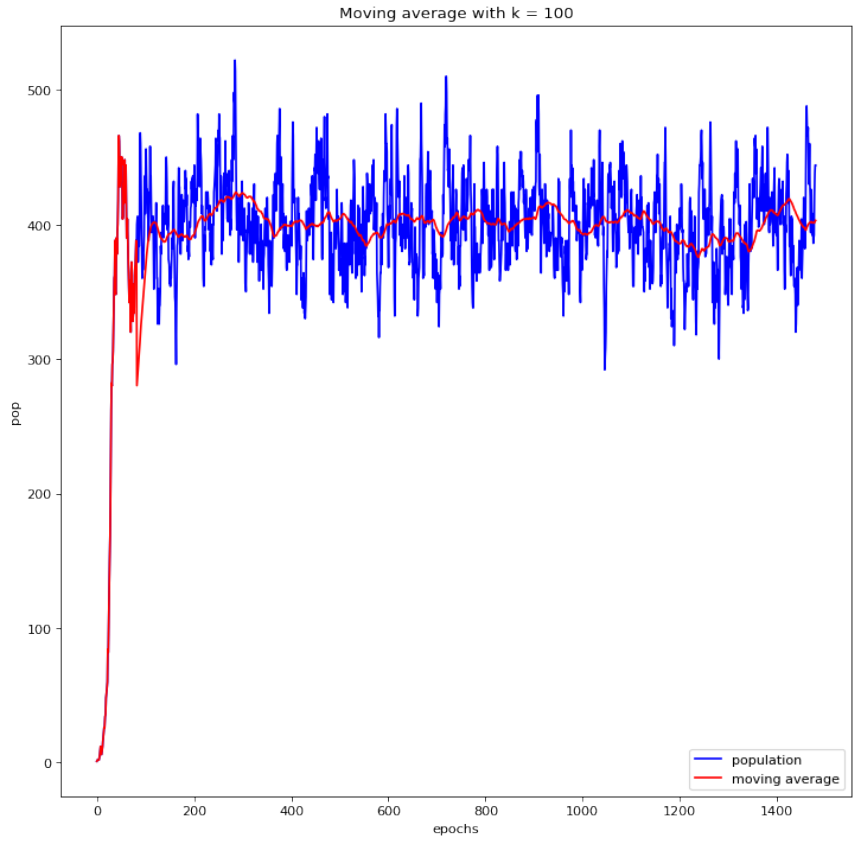

Рис. 3.4: Применение простого скользящего среднего с размером окна 100

$\hat{p}_t$ - сглаженное значение численности популяции в момент времени t.

Изначально простое экспоненциальное сглаживание было реализовано с помощью модели SimpleExpSmoothing из библиотеки statsmodels, а оптимальный параметр $\alpha = 0.72$ подобран с помощью встроенного метода. Однако такое высокое значение практически полностью устранило сглаживание, а вид графика почти не изменился. После этого сглаживание было написано вручную, и оптимальный параметр подбирался с помощью минимизации MAE/MSE по всем значениям $\alpha \in (0, 1)$ с шагом 0.05. Этот подход также оказался неудачным, так как в MAE (Mean Absolute Error) и MSE (Mean Squared Error) берутся абсолютные значения разности между сглаженной численностью популяции и исходным значением, из-за чего график все еще плохо сглаживается. Новый оптимальный коэффициент составил 0.68 (Рисунок 3.5) и был так же отвергнут, после чего поиск коэффициента шел визуальным методом по уже сгенерированным данным с разными видами зависимости численности популяции от времени. Изначально был выбран $\alpha = 0.4$ - рисунок 3.6, и было решено проводить 6 последовательных сглаживаний с этим коэффициентом, так как это давало лучший результат по сравнению с применением сглаживания один раз или применения сглаживания с более низким коэффициентом, а большее количество повторений не улучшало результат, однако приводило к более значительным ошибкам в зоне роста численности популяции.

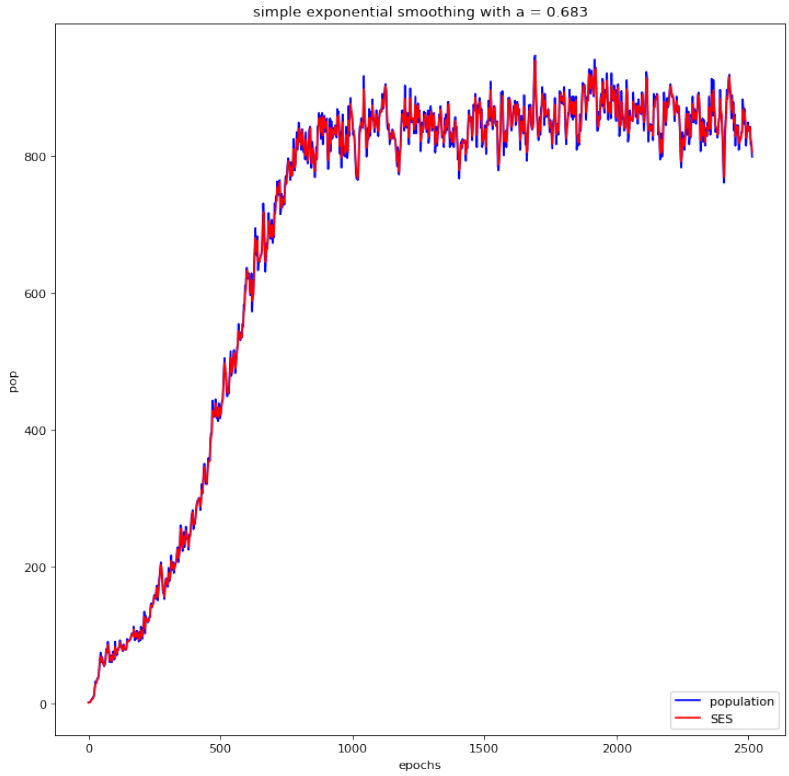

Рис. 3.5: Применение простого простого экспоненциального сглаживания с $\alpha = 0.683$

В ходе генерации симуляций это значение было снижено до $\alpha = 0.3$ с 6 последовательными сглаживания, так как это позволило раньше останавливать симуляции. Попытка применить меньшие значения α окончилась тем, что симуляции завершались, не успевая выйти на плато. Все сглаживания далее построены с этим коэффициентом.

3.2.3 Двойное экспоненциальное сглаживание

Кроме того, было применено двойное экспоненциальное сглаживание. Это расширение простого экспоненциального сглаживания, достигающееся введением второго параметра β, который отвечает за силу влияния тренда: чем выше β, тем сильнее влияние. Так, формула для двойного сглаживания:

$$\begin{aligned} b_0 &= p_1 - p_0, \quad \hat{p}_0 = p_0 \\ \hat{p}_{t+1} &= \alpha p_{t+1} + (1-\alpha)(\hat{p}_t + b_t) \\ b_{t+1} &= \beta(p_{t+1} - p_t) + (1-\beta)b_t, \quad \beta \in [0,1) \end{aligned} \tag{6}$$

Лучший результат был получен при $\beta = 0$ - рисунок 3.7. Это можно объяснить тем, что в данных не один и не два тренда: значение каждой следующей точки зависит сразу от того, что происходит со каждым организмом в популяции. Итак, было выбрано простое экспоненциальное сглаживание с $\alpha = 0.3$, которое последовательно применялось 6 раз.

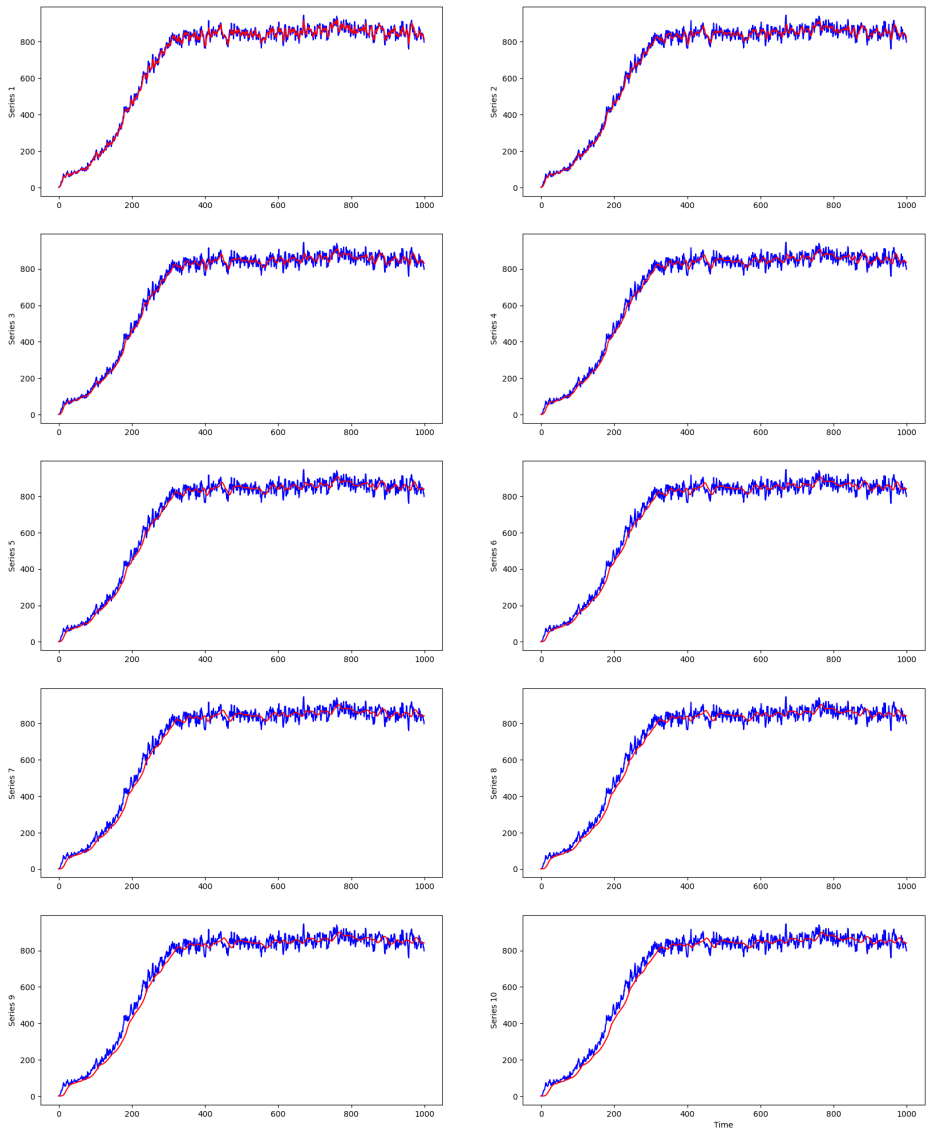

Рис. 3.6: Последовательное использование простого простого экспоненциального сглаживания с $\alpha = 0.4$ 1-10 раз

3.3 Поиск момента выхода на плато

Разобьем данные о численности популяции на несколько срезов одинаковой длины (например, наблюдения 1-1000, 1001-2000, и так далее). Для каждого среза посчитаем его дисперсию. В предыдущих работах научной группы [5] было показано, что дисперсия срезов падает с выходом на плато, однако существует часть графиков, где эта тенденция выражена не так сильно. Однако почти для всех графиков можно выделить три зоны: зона роста, зона плато и зона колена: рисунок 3.8.

Цель этой части работы - подобрать порог соотношения дисперсий отрезков с помощью которого можно убедиться, что график вышел в зону плато и остановить симуляцию, сэкономив вычислительные ресурсы. Для начала рассмотрим возможные варианты длины одного среза, а затем определим начальный порог соотношения дисперсий, свидетельствующий о выходе

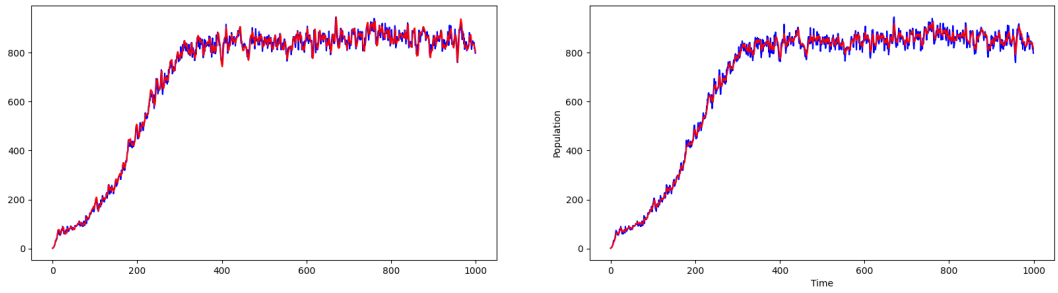

Рис. 3.7: Применение двойного экспоненциального сглаживания, $\alpha = 0.4$: слева - $\beta = 0.3$, справа - $\beta = 0$

на плато. Далее по группе графиков, где выход на плато не был обнаружен, проведем дополнительное исследование возможного соотношения дисперсий отрезков. Результаты можно увидеть в **Приложении В**. С помощью результатов раздела модифицируем алгоритм поиска момента выхода на плато для одномерных симуляций, переработав его для двумерных симуляций.

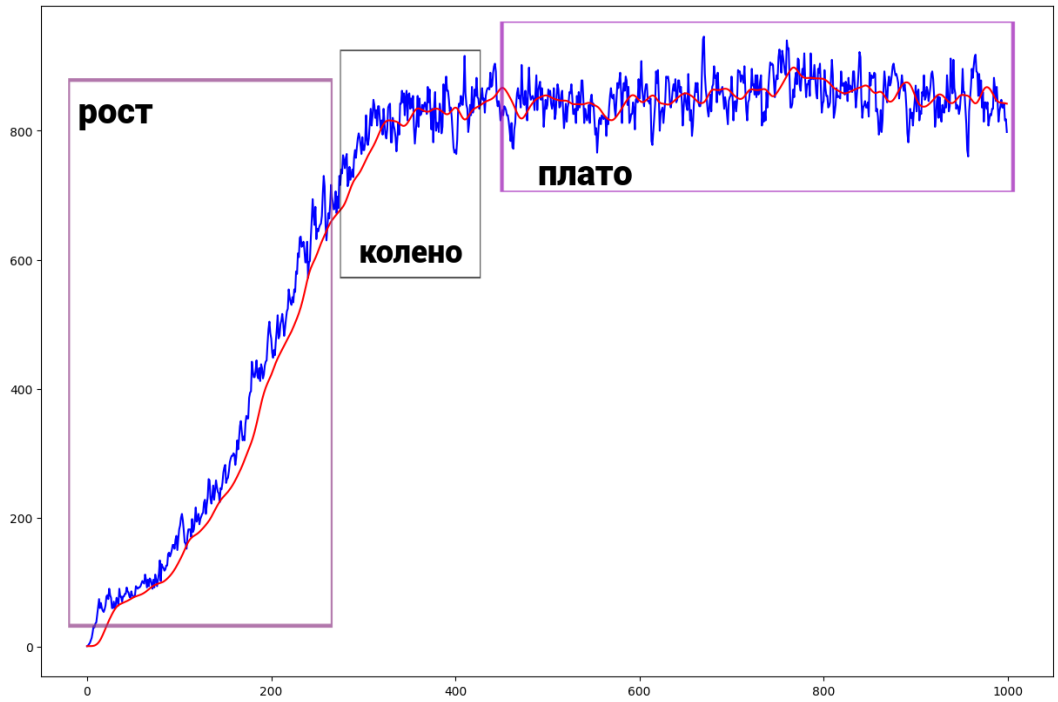

Рис. 3.8: Зоны графика численности популяции: плато, колено, рост

3.3.1 Выбор длины срезов

Для подбора использовались длины срезов 100, 250, 500, 1000 и 1500 эпох. Соотношения дисперсий на срезах такой длины для одной из симуляций представлены на рисунках 3.9 - 3.12. Уменьшение дисперсий срезов заметно при выборе любой длины среза, однако слишком маленькая длина среза может привести к ложной остановке симуляции на выбросе. По-

этому было принято решение оставить значения длин срезов, использующиеся в алгоритме для одномерных симуляций [3]:

$$size_1 = max(epochs//5, 1000), \tag{7}$$

$$size_2 = max(epochs//10, 500), \tag{8}$$

где epochs - количество эпох, заданное в симуляции. Если для срезов хотя бы одной длины достигается порог выхода на плато, симуляция останавливается.

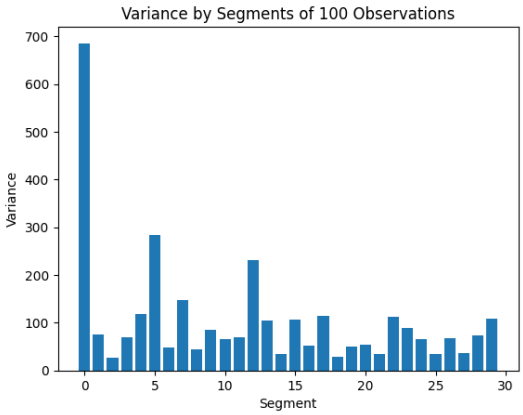

Рис. 3.9: Дисперсии на срезах длины 100

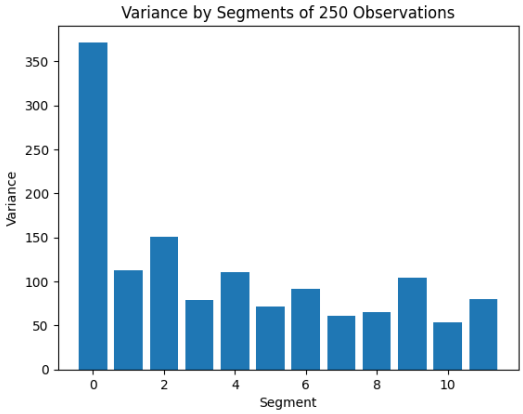

Рис. 3.10: Дисперсии на срезах длины 250

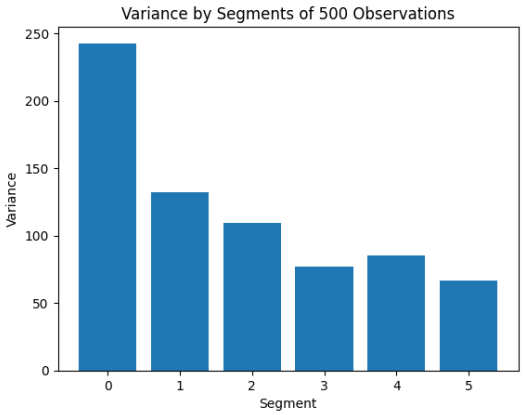

Рис. 3.11: Дисперсии на срезах длины 500

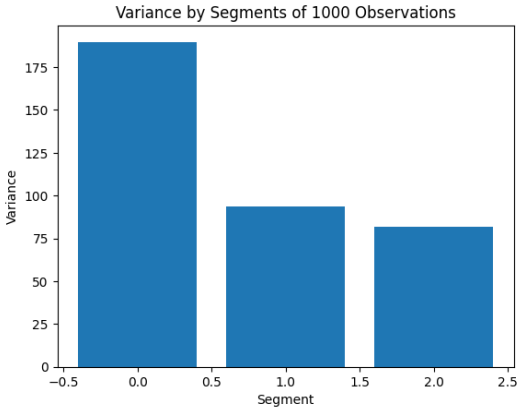

Рис. 3.12: Дисперсии на срезах длины 1000

3.3.2 Исследование соотношения дисперсий срезов

Дополнительно создан датасет с 561 популяцией, численность которой не вышла на плато за 3000 эпох. Для начала рассмотрим Таблицу 3.2 с характеристиками этих симуляций. Все они обладают довольно высоким значением смертности от конкуренции, но при этом у них довольно высокая рождаемость, позволяющая это компенсировать. Такие симуляции зашумлены и то последовательно растут, то убывают, что видно на Рисунке 3.13

Таблица 3.2: Описательные статистики симуляций, не вышедших на плато

index	dd	death_r	area_length_y	b	sd_b	sd_d	d	final_pop
count	561.0	561.0	561.0	561.0	561.0	561.0	561.0	561.0
mean	0.26	2.22	10.86	0.48	0.76	0.74	0.23	103.31
std	0.11	0.50	5.76	0.14	0.17	0.17	0.12	87.80
min	0.11	1.50	5.00	0.15	0.50	0.50	0.10	10.00
25%	0.21	1.95	5.00	0.35	0.65	0.65	0.10	32.00
50%	0.21	2.40	11.00	0.45	0.80	0.80	0.20	82.00
75%	0.31	2.85	16.00	0.55	0.95	0.95	0.30	142.00
max	0.41	2.85	25.00	0.86	0.95	0.95	0.60	436.00

Для набора симуляций были вычислены 1 и 2 максимумы отношений дисперсий сосед-

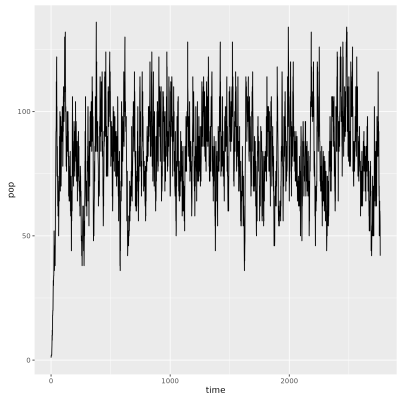

Рис. 3.13: Пример графика симуляции, не вышедшей на плато

них срезов для размеров срезов, заданных формулами 7 и 8. Изначально в качестве порога стояло значение 10, но как видно, для большинства первых максимумов из выборки оно не было достигнуто. В связи с этим на исследуемой области можно несколько понизить порог выхождения на плато до 7.

Таблица 3.3: Первые максимумы отношений дисперсий соседних срезов

Статистика	**Значение**
Среднее	4.54
Стандартное отклонение	3.27
Минимум	1.00
25-й перцентиль	2.04
50-й перцентиль	3.36
75-й перцентиль	6.32
Максимум	18.15

Таблица 3.4: Вторые максимумы отношений дисперсий соседних срезов

Статистика	**Значение**
Среднее	1.49
Стандартное отклонение	0.38
Минимум	1.02
25-й перцентиль	1.21
50-й перцентиль	1.40
75-й перцентиль	1.65
Максимум	2.98

3.3.3 Модификация алгоритма поиска момента выхода на плато

Переработанный алгоритм был протестирован на группе тестовых симуляций в **приложении C**, код самого алгоритма представлен в **приложении D**

4 Машинное обучение

4.1 Сборка датасета

При создании датасета исключались эксперименты, где популяция скорее всего вымрет. Например, максимальная смертность от конкуренции ограничивалась 0.41. Также рассматривались только значения, где b > d. Для всех симуляций задавалась начальная популяция 1 и максимальное количество эпох 3000, а в качестве области выбирался квадрат с стороной от 1 до 25. Всего было сгенерировано 26526 симуляций, однако для 10000 из них была сгененрирована на другом компьютере, где не сохранились данные о том, вышла на численность популяции на плато или нет. Поэтому в итоговом датасете использовалось 15494 симуляции, точно вышедших на плато. Для создания датасета был написан скрипт: **приложение Е**. В датасет вошли лишь сами параметры симуляций и целевая переменная - значение численности популяции на плато. Все остальные значения численности в датасете не использовались.

Ниже приведена таблица 4.1 с распределением параметров в датасете. Значения округлены до 2 знаков после запятой.

Таблица 4.1: Распределение параметров в итоговом датасете

index	dd	area_length_y	b	sd_b	sd_d	d	final_pop
count	15446.00	15446.00	15446.00	15446.00	15446.00	15446.00	15446.00
mean	0.19	9.39	0.60	0.72	0.72	0.31	1140.29
std	0.14	7.57	0.20	0.17	0.17	0.18	3863.20
min	0.01	2.00	0.15	0.50	0.50	0.10	0.00
25%	0.11	3.00	0.45	0.50	0.50	0.20	0.00
50%	0.21	5.00	0.65	0.65	0.65	0.30	28.00
75%	0.31	16.00	0.75	0.80	0.80	0.40	416.00
max	0.41	25.00	0.96	0.95	0.95	0.70	44592.00

Отдельно рассмотрим значение целевой переменной в таблице 4.2Как видно, в датасете мало примеров, когда популяция вымирает. Кроме того, мало и высоких значений: 90 процентов данных меньше, чем 2000.

Таблица 4.2: Квантили итоговой численности

Квантиль	Итоговая численность
0.25	0.0
0.50	28.0
0.75	416.0
0.90	2003.5
0.99	21680.7

И, наконец, матрица корреляции между параметрами - Рисунок 4.1. Видно, что на значение численности на плато сильнее всего влияют смертность от конкуренции (отрицательно) и площадь зоны обитания (положительно).

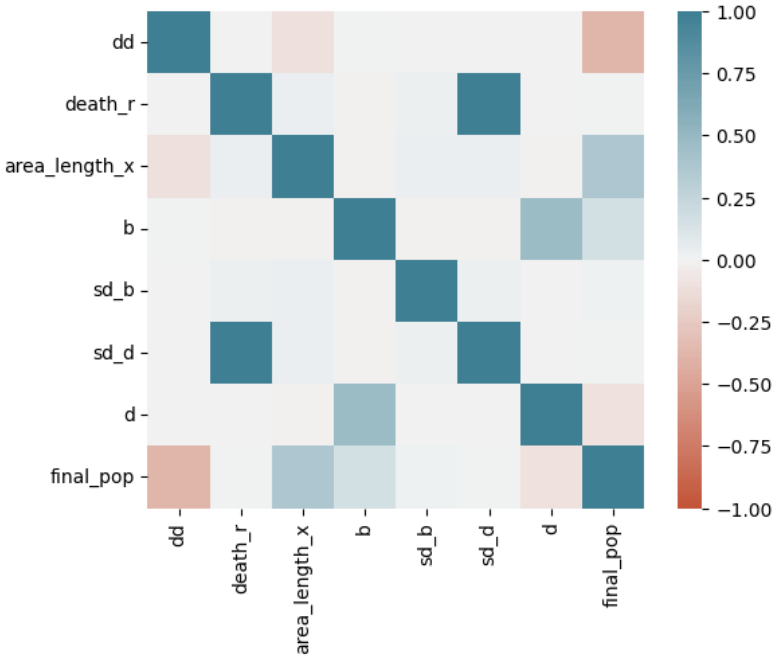

Рис. 4.1: Матрица корреляции между параметрами распределения

4.2 Подготовка к обучению

Для однозначности и воспроизводимости результатов в блокноте был зафиксирован random seed = 1337. Далее данные были нормализованы с помощью Standard Scaler из библиотеки sklearn, так как часть параметров отличается на несколько порядков, что может ухудшить результаты обучения.

В качестве метрики использовалась RMSE (Root Mean Square Error). Ее преимущество по сравненению с MSE состоит в том, что RMSE выдает среднюю ошибку в тех же единицах, что и целевая переменная, поэтому такую ошибку легче интерпретировать.

Для проверки качества каждого алгоритма использовалась кросс-валидация, а именно метод k-fold. Данные были разбиты на 4 равные частей, и каждый раз 1 из них использовалась для тестирования, а остальные 4 для обучения. В качестве итогового значения ошибки выбиралось среднее из всех ошибок. Все модели представлены в **приложении F**.

4.3 Линейная регрессия

В качестве простейшей модели сразу была применена линейная регрессия. RMSE составила 3246, что является очень большим результатом. Значит, эта модель не подходит для решения задачи.

4.4 Случайный лес

Далее был обучен случайный лес, применяющийся и для задачи регрессии. Он основан на комбинации нескольких решающих деревьев в единый ансамбль.

Каждое решающее дерево в случайном лесу строится на основе случайной подвыборки обучающих данных, а также случайного подмножества признаков. Это делается для уменьшения корреляции между деревьями и увеличения разнообразия моделей в ансамбле. Когда необходимо сделать прогноз, каждое дерево в лесу вносит свой собственный вклад, и результаты агрегируются для получения окончательного прогноза, в задаче регрессии результаты просто усредняются.

Для минимизации ошибки был произведен перебор параметров, RMSE с этими параметрами составила 307

4.5 Градиентый бустинг

Наконец, был применен градиентный бустинг.

Градиентный бустинг - это ансамблевый метод, который комбинирует несколько слабых моделей обучения, в нашем случае решающих деревьев для создания более мощной модели. Основная идея заключается в последовательном обучении моделей, каждая из которых исправляет ошибки предыдущей модели. В отличие от случайного леса модели скорелированы: каждая новая модель в градиентном бустинге строится таким образом, чтобы минимизировать остаточные ошибки предыдущих моделей. В конечном итоге получается ансамбль моделей, который обладает высокой предсказательной способностью.

В итоге я получила результат: RMSE = 312.

4.6 Глубокое обучение

Для следующей части работы была реализована полносвязная нейронная сеть.

Полносвязная нейронная сеть состоит из нескольких слоев нейронов, которые обрабатывают входные данные и генерируют выходные результаты. Каждый нейрон в одном слое

соединен с каждым нейроном в следующем слое. Это означает, что каждый нейрон входного слоя связан с каждым нейроном скрытого слоя, а каждый нейрон скрытого слоя связан с каждым нейроном выходного слоя.

Каждая связь между нейронами имеет свой вес, который определяет влияние данной связи на результаты нейронной сети. Веса этих связей обучаются в процессе обучения сети на основе примеров данных.

Функция активации применяется к выходу каждого нейрона и определяет, будет ли активирован нейрон или нет. Она обычно включает нелинейное преобразование, позволяющее сети моделировать сложные нелинейные зависимости между данными.

Архитектура нейросети подбиралась исходя из задачи, она состоит из 4-ех линейных слоев, после каждого из которых идет функция активации ReLU за исключением последнего, количество нейронов во входном слое равно количеству признаков в датасете, каждый внутренний слой состоит из 128 нейронов.

Нейросеть обучалась на минимизацию MSE при помощи оптимизатор Adam с learning rate = 0.001, всего было 10 эпох обучения. Зависимость MSE от эпохи обучения представлена на рисунке 4.2

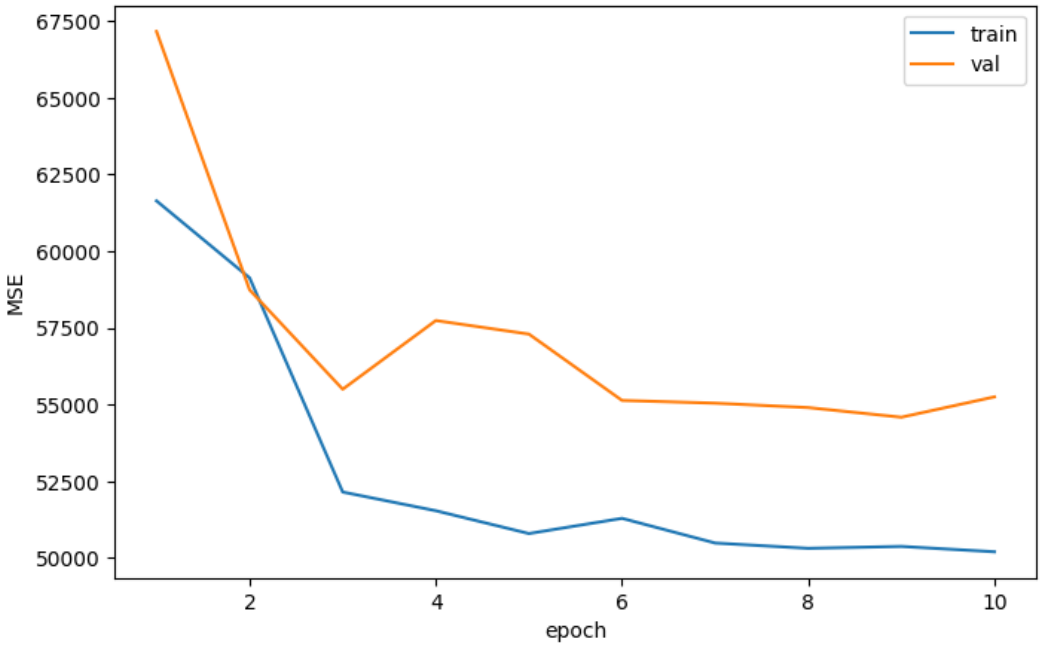

Рис. 4.2: Зависимость MSE от эпохи обучения нейронной сети

По итогу обучения полносвязаная нейросеть показала лучший результат из всех ранее тестируемых моделей, RMSE оказался равным порядка 235, что вполне неплохо исходя из масштаба входной переменной.

5 Итоги работы

В рамках работы были изучены двумерные симуляции модели Дикмана-Лоу,а также написан пример для их запуска. Далее с помощью сглаживания исследованы возможные значения дисперсии срезов численности популяции и подобрано пороговое соотношение, при котором симуляцию следует остановить. Это позволило сэкономить вычислительные ресурсы при генерации примеров, так как при нахождении момента выхода на плато симуляция останавливается. Был собран датасет из двумерных симуляций с начальной численностью популяции 1. На основании этих данных обучены классические модели: линейная регрессия, случайные деревья, градиентный бустинг. Наилучший результат показали случайные деревья, ранее не использовавшийся для предсказания одномерных данных. Также обучена нейросеть, подобрана ее оптимальная архитектура. Наилучшее итоговое значение RMSE составило 221, что является хорошим результатом, учитывая небольшой объем датасета и порядок целевой переменной.

Дальше можно увеличивать и расширять датасет: взять примеры с разной начальной численностью популяции, более крупной площадью, в качестве области обитания выбирать прямоугольник, а не квадрат.

Список литературы

[1] Law R Dieckmann U. “Dieckmann U., Law R. Relaxation projections and the method of moments”. В: *Cambridge University Press* (2000).

[2] Law R Dieckmann U. “Moment approximations of individual-based models”. В: *Cambridge University Press* (2000).

[3] Соколова Д.Д. “Стохастические симуляции биологической модели стационарных сообществ”. В: *Курсовая работа* (2023).

[4] А.А.Никитин Е.Г.Галкин В.К. Зеленков. “Компьютерные симуляции и численные методы в двухвидовой модели пространственных сообществ”. В: *International Journal of Open Information Technologies* (2019).

[5] Михайлова К.Д. “Исследование стационарной модели биологических сообществ”. В: *Курсовая работа* (2022).

Приложение А

Код для построения сглаживаний

https://github.com/ann-vasileeva/Coursework23/blob/main/smoothing.ipynb

Приложение В

Исследование возможного соотношения дисперсии отрезков

https://github.com/ann-vasileeva/Coursework23/blob/main/dispersion.ipynb

Приложение C

Тестирование обновленного алгоритма поиска момента выхода на плато

https://github.com/ann-vasileeva/Coursework23/blob/main/plato.ipynb

Приложение D

Обновленный код алгоритма поиска момента выхода на плато

https://github.com/ann-vasileeva/Coursework23/blob/main/runsimulation.R

Приложение Е

Скрипт для создания датасета

https://github.com/ann-vasileeva/Coursework23/blob/main/createdata.py

Приложение F

Подбор оптимальной модели для обучения

https://github.com/ann-vasileeva/Coursework23/blob/main/ml.ipynb