НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 007.52:519.6:539.3:681.3

До захисту допущено
Завідувач кафедри ММСА
Оксана ТИМОЩУК
«___» _____ 2021 р.

Магістерська дисертація

на здобуття ступеня магістра за спеціальністю 124 Системний аналіз
на тему: «Формування множини Парето в задачах
пошуку раціонального компромісу»

Виконав: студент ІІ курсу, групи КА-01мп
Шарапов Євгеній Олександрович _____

Науковий керівник: професор кафедри ММСА,
д.т.н., проф. Панкратова Н.Д. _____

Рецензент:
доцент кафедри автоматизації
теплоенергетичних процесів
КПІ ім. Ігоря Сікорського
к.т.н., доц. Голінко І.М. _____

Засвідчую, що в цій магістерській дисертації
немає запозичень з праць інших авторів
без відповідних посилань
Студент _____

Київ
2021

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти — другий (магістерський)
Спеціальність (спеціалізація) — 124 «Системний аналіз» («Системний аналіз і управління»)

ЗАТВЕРДЖУЮ
Завідувач кафедри ММСА
Оксана Тимощук
«___» _____ 2021 р.

ЗАВДАННЯ

на магістерську дисертацію студенту Шарапову Євгенію Олександровичу

1. Тема дисертації: «Формування множини Парето в задачах пошуку раціонального компромісу», науковий керівник дисертації Панкратова Наталія Дмитрівна, д.т.н., проф., затверджені наказом по університету від «02» листопада 2021 р. № 3651-с

2. Термін подання студентом дисертації: 13 грудня 2021 року

3. Об’єкт дослідження: процеси складних систем із великою кількістю внутрішніх та зовнішніх показників

4. Предмет дослідження: методи формування множини Парето за узгодженням областей значень та визначень

5. Перелік завдань, які потрібно розробити:

1) дослідити стан проблеми формування множини Парето для складних технічних виробів;

2) віднайти методи узгодження внутрішніх та зовнішніх показників;

3) на основі отриманих результатів розробити метод машинного навчання, що може бути застосований до формування множини Парето;

4) розробити стартап проєкт за результатами проведених досліджень.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:

1) ілюстративний матеріал існуючих алгоритмів;

2) схеми та таблиці методів узгодження внутрішніх та зовнішніх показників;

3) таблиці у розділі стартап проекту;

4) графіки навчання компонент архітектури.

7. Орієнтовний перелік публікацій:

Губарев В.Ф. Оценка решений переопределенных СЛАУ с неточно заданной правой частью / Губарев В.Ф., Шарапов Е.А. //Кибернетика и системный анализ: міжнародний науково-теоретичний журнал. — Київ: Інститут кібернетики ім. В. М. Глушкова НАН України. — 2021. — № 2. — С. 96—109. — ISSN 0023-1274 (Шифр К499224528/2021/2). - ISSN 0023-1274

Шарапов Є.О. Методи узгодження внутрішніх та зовнішніх показників виробу в задачах формування множини Парето / Шарапов Є.О. // Праці XIX міжн наук.-практ. конф. «Математичне та програмне забезпечення інтелектуальних систем» (17—19 листопада, 2021, Дніпро). — С. 9—10

8. Дата видачі завдання: 01 вересня 2021 року

Календарний план

№/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації
1	Огляд літератури за тематикою	01.09.2021—07.09.2021
2	Збір матеріалів з проблеми магістерської дисертації	08.09.2021—10.09.2021
3	Вивчення матеріалів	11.09.2021—26.09.2021
4	Аналіз існуючих методів розв’язування задачі	27.09.2021—01.10.2021
5	Створення програмного продукту	02.10.2021—25.10.2021
6	Тестування та налаштування	26.10.2021—10.11.2021
7	Оформлення звіту	11.11.2021—03.12.2021

Студент Євгеній ШАРАПОВ

Науковий керівник дисертації Наталія ПАНКРАТОВА

РЕФЕРАТ

ШТУЧНІ НЕЙРОННІ МЕРЕЖІ, МНОЖИНА ПАРЕТО, МІШАНЕ ЛІНІЙНЕ ПРОГРАМУВАННЯ, СЕГМЕНТАЦІЙНІ МОДЕЛІ, СИСТЕМНИЙ АНАЛІЗ

Магістерська дисертація 103 с., 31 рис., 19. табл, 1 додаток, 17 джерел

Об’єктом досліджень є процеси у складних системах із великою кількістю вхідних та вихідних параметрів.

Предметом дослідження є методи формуваня множини Парето, узгодження областей значень та визначень.

Мета дослідженя:

1. Розробити та дослідити методи узгодження областей значень та визначень для функцій за дискретно заданою вибіркою.
2. Створити модель формування множини Парето.

Новизна: Використання нейронних мереж для формувая функціональних залежностей із подальшим формуванням множини Парето та корекції за допомогою методів узгодження областей значень та визначень.

В роботі проводиться ретельна побудова методів узгодження областей значень та визначень для функцій, за дискретно заданою вибіркою. Для цього використовується апарат штучних нейронних мереж. В результаті отриманий інструментарій знаходить застосування у корекції множини Парето, яка формується із використанням комбінації нейронних мереж, що включають в себе моделі сегментації.

В результаті було створено модель побудови множини Парето для системи функцій за дискретно заданою вибіркою, що дозволяє знаходити компроміс під час прийняття рішення щодо моделювання складного виробу.

ABSTRACT

ARTIFICIAL NEURAL NETWORKS, PARETO SET, MIXED LINEAR PROGRAMMING, SEGMENTATION MODELS, SYSTEM ANALYSIS

Master's dissertation 103 p., 31 fig., 19. table, 1 appendix, 17 sources

The object of research is complex systems with a large number of input and output parameters.

The subject of the research is the methods of reconstruction of the Pareto set, coordination of the domains of arguments and values.

The purpose of the study:

1) Develop and investigate methods for reconciling arguments domain and values domains for tabular functions.

2) Form a Pareto domain recovery model.

Novelty: The use of neural networks to restore functional dependencies with subsequent forecasting of the Pareto area and correction using methods of matching the areas of values and definitions.

The work carefully constructs methods for reconciling the areas of values and arguments for functions that they are given by tabular data. The approach of the artificial neural networks is used for this purpose. The resulting toolkit is used in the correction of the Pareto set, which is reconstructed using a combination of neural networks that include segmentation models.

As a result, a model for constructing a Pareto set was created for a system of functions defined in tabular form, which allows to find a compromise when deciding on the modeling of a complex product.

ЗМІСТ

ВСТУП

Виготовлення довільного продукту передбачає встановлення взаємозалежності певних компонент, що мають внутрішні та зовнішні показники. До перших відносять такі, які вважаються константними на момент проектування та постають в якості аргументів та вхідних даних для других, які визначають частіше за все вихідні характеристики виробу. Прикладом перших можуть слугувати ,наприклад, розміри функціональних запчастин - коленвалів, поршнів тощо, на відміну від котрих другі можуть становити певні показники - швидкість чи потужність двигуна та багато інших.

Нескладно побачити, що велика кількість сучасних проблем можна описати в такій постановці задачі. Задачі проектування, моделювання чи інтегрування функціональних модулей чи елементів. На практиці часто застосовують наступний підхід.

Таким чином все більш наочними стають проблеми узгодження внутрішніх та зовнішніх показників технічних систем, встановлення взаємозв’язків між котрими стає архіважливим на ринку.

Метою даної роботи є формалізація та дослідження існуючих методів побудови такої взаємовідповідності методом пошуку множини Парето. Порівняння та окреслення можливих областей застосування.

Об’єктом дослідження є процеси, що вимагають узгодження внутрішніх та зовнішніх показників.

Предметом дослідження є методи побудови області Парето.

Перший розділ включає в себе огляд існуючої теоретичної бази, що охоплює класичні методи та підходи до вирішення поставленої задачі. Другий розділ роботи ставить за мету опис запропонованих методів до розв’язку поставленої проблеми, надання рекомендацій до використання, описує переваги та недоліки кожного. Третій розділ розглядає застосування кожного з описаних методів у проекції на реальні проблеми та задачі. Четвертий розділ

наводить приклад стартапу за поданою тематикою дослідження.

РОЗДІЛ 1: ЗАДАЧА УЗГОДЖЕННЯ ОБЛАСТЕЙ ЗНАЧЕНЬ-ВИЗНАЧЕНЬ ТА ПОШУКУ МНОЖИНИ ПАРЕТО

1.1 Передумови виникнення проблеми

Проблеми узгодження внутрішніх та зовнішніх показників складних систем виникає повсемісно у контексті сучасних реалій. Даний розділ ставить за мету описання передумов виникнення даної задачі в історичній площині.

Першочергово варто окреслити вагомий вклад основоположника науки кібернетики - Глушкова В. М., за керівництвом якого було розроблено першу в світі ЕВМ “МИР 1”. Одна з перших в світі персональних ЕОМ, що випускалася серійно і призначалася для використання в навчальних закладах, інженерних бюро та наукових організаціях. Володіла рядом унікальних особливостей, таких як апаратно реалізована машинна мова, близька за можливостями до мов програмування високого рівня, розвинене математичне забезпечення. Саме видатним радянським вченим була вперше поставлена проблема узгодження вхідних та вихідних параметрів певного виробу [5]

Важливою першопричиною слугувала існуюча в радянському союзі на той час загроза військового конфлікту та фактичні перегони спорядження, що вимагали створення та розроблення високоякісних засобів [1]. Не менш важливим виявлялися науково-технологічні передумови, що ставили задачу створення складних алгоритмічних машин, що були би здатні виконувати величезну низку процедур та операцій. Головною особливістю Холодної війни була гонка озброєнь між державами - членами Варшавського договору і НАТО. Незважаючи на свою руйнівну, вона привела до суттєвих наукових відкриттів у багатьох технологічних і військових областях. Для гонки озброєнь була характерна підвищена міжнародна напруженість і нестабільність, постійні політичні скандали, постійні випробування нових видів зброї і використання

військової сили як основного аргументу в політичних питаннях. Однак, незважаючи на це, багато в чому завдяки руйнівним продуктам гонки озброєнь, Холодна війна так і не переросла в гарячу під час численних криз і локальних конфліктів за участю наддержав.

Якщо розмірковувати в руслі сучасних проблем та задач, що ставить перед фахівцями сучасність, то вкрай важливим стають створення сучасних засобів, таких, як наприклад, складні засоби пересування.

Розглянемо для ілюстративного прикладу задачу побудови екскаватора, що має бути здатним працювати в екстремальних умовах таких як арктична тундра. Оскільки робота механізму вимагає наявності достатніх потужностей та тривалої роботи, то природним є обрати за вихідні змінні тривалість гарантованої безперебійної роботи двигуна, потужність двигуна та швидкість виконання роботи. Варто помітити, що зазначені вихідні змінні є конфліктними, так, намагаючись збільшити потужність двигуна, конструктор автоматично вдається до збільшення ризику перебою в роботі двигуна як за рахунок витрат палива так і за рахунок більш довготривалого зношення. Ті самі міркування наводять на негативну кореляцію між швидкістю виконання роботи та тривалістю роботи. Велику роль в даному прикладі грають керовані(вхідні) змінні, за які обирають ті показники, що можуть змінюватися в період проектування засобу, ці показники можуть бути, наприклад, розмірами стержнів, розмір баку, кількість свічок у двигуні чи розмір сопла. Наявність некерованих змінних також може відігравати другорядну роль, адже вклад таких часто зараховують при моделюванні на наявність певних некорельованих шумів у вихідних величинах.

Узгодження вихідних та вхідних характеристик тепер виявляється ключовим. Воно дозволяє дати відповідь на такі важливі питання, як чутливість вихідних характеристик відносно вхідних таким чином, що при знаходженні бажаної зміни вихідних характеристик, віднайти допустиму зміну вхідних показників (фактичне встановлення неперервності заданої функції відносно

вхідних параметрів), а також область Парето конфліктуючих задач (вхідних змінних).

Декілька важливих зауважень щодо сучасних застосувань варто навести через те, що великі системи набувають все більшу кількість керованих параметрів та не меншу кількість вихідних параметрів. На сьогоднішній день поняття “Big Data” набуває все більшого значення, технології стають схильними до використання хмарних обчислень.

Принцип роботи технології big data заснований на максимальному інформуванні користувача про який-небудь предмет або явище. Завдання такого ознайомлення з даними - допомогти зважити всі «за» і «проти», щоб прийняти вірне рішення. В інтелектуальних машинах на основі масиву інформації будується модель майбутнього, а далі імітуються різні варіанти і відслідковуються результати. Це допомагає внести вклад в створення. Чим більше ми знаємо про конкретний предмет або явище, тим точніше осягаємо суть і можемо прогнозувати майбутнє. Знімаючи і обробляючи потоки даних з датчиків, інтернету, транзакційних операцій, компанії можуть досить точно передбачити попит на продукцію, а служби надзвичайних ситуацій запобігти техногенним катастрофам. Наведемо кілька прикладів поза сферою бізнесу і маркетингу, як використовуються технології великих даних. Одним із основних напрямків є Машинне Навчання, як невід’ємна компонента сучасних передових систем. Воно дозволяє розв’язувати складні проблеми, які в багатьох випадках не допускають простого алгоритмічного розв’язку, що передбачає певну детермінованість дій та процедур.

В такій постановці задачі стає можливим створення сучасних засобів взаємодії між кінцевим користувачем та процедурою, що здатна розраховувати необхідну область Парето для заданої системи. Важливість такої інтеграції полягає в тому, що фактична взаємодія пристроїв, що складають кіберфізичну систему на сьогоднішній день, складає стандарт інтернету речей в тому сенсі, що сервіси та послуги, що оперують у хмарному середовищі пов’язані один з

одним та дозволяють ефективно встановити зв’язок між компонентами складної обчислювальної системи. Це обумовлює актуальність даної роботи, а також методів, що можуть бути використані для встановлення даної відповідності, їх ефективність залежить від складності сімейства наближуючих функцій та інших властивостях задачі. В наступному розгляді буде розглянуто щонайзагальнішу постановку задачі узгодження внутрішніх та зовнішніх показників.

1.2 Поняття Кіберфізичної системи

Поняття кіберфізичної системи є передовим концептом та провідним поняттям 21-го сторіччя. Першочергово вона передбачає інтеграцію обчислювальних ресурсів в фізичні сутності будь-якого виду, включаючи біологічні та рукотворні об'єкти. У кіберфізичних системах обчислювальна компонента розподілена по всій фізичній системі, яка є її носієм, і синергетично пов’язана з її складовими елементами. Основними елементами такої системи є, як нескладно здогадатися, фізична складова, що може бути представлена у вигляді певного носія, наприклад веб-сервера, чи хмарного середовища (рис. 1.1):

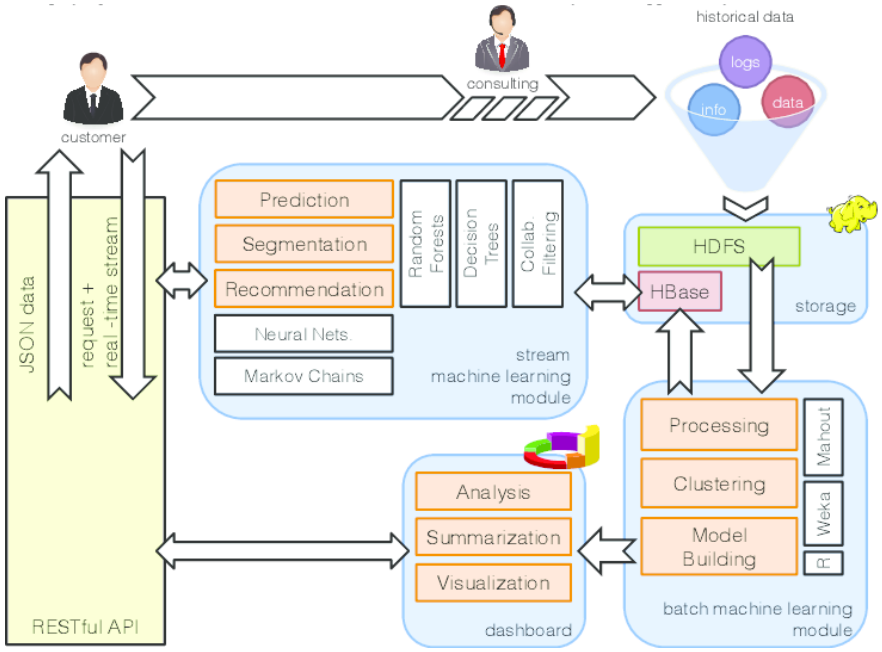

Рисунок 1.1 - Приклад кіберфізичної системи

Можна перерахувати ключові технологічні тенденції, що лежать в основі кіберфізіческіх систем. Вони вже використовуються в окремих сферах, але, будучи інтегрованими в єдине ціле, вони змінюють існуючі відносини між виробниками, постачальниками і покупцями, а також між людиною і машиною[2]:

1.Великі дані і аналітика - збір і всебічна оцінка даних з різних джерел стають стандартом для прийняття рішень в режимі реального часу.

2.Моделювання та симулятори - інженери вже використовують 3D-моделювання на стадії проектування продуктів або процесів. В майбутньому технології великих даних дозволять використовувати різні симулятори в режимі реального часу. Наприклад, на стадії виробництва оператор зможе віртуально змоделювати фізичний процес з урахуванням наявної сировини і людей, тим самим знизити час налаштування обладнання та підвищити якість.

3.Інтернет речей [2-4] - показники датчиків і сенсорів зазвичай потрапляють в централізовану систему управління виробничим процесом, і вже на цьому рівні приймаються рішення. Надалі можливості, які надають

вбудовані системи, дозволять пристроям спілкуватися один з одним і децентралізувати прийняття рішень. Наприклад, можна використовувати радіочастотні мітки для напівфабрикатів - автоматизована виробнича лінія, прочитавши мітку, сама прийме рішення (в реальному часі), яку операцію застосувати до того чи іншого напівфабрикату.

4.Доповнена реальність - технологія знаходиться в початковій стадії свого розвитку, але в майбутньому дозволить працівникам прискорити прийняття рішень. Наприклад, працівник може отримати інструкцію, як полагодити або замінити зламану деталь у виробничій системі, коли він на неї дивиться через окуляри доповненої реальності.

5.Хмарні обчислення - потрібна більш глибока системна інтеграція, як горизонтальна між постачальниками і клієнтами, так і вертикальна між різними функціями та операціями. Необхідне Створення платформи для спільної роботи і взаємообміну даними між територіально-розподіленими партнерами, що і дозволяють хмарні технології.

Зазначимо, що можливість «зробити життя людей краще і простіше» за допомогою цих систем відмінно можна проілюструвати як раз на прикладі «розумних» міст. Сінгапур вже неодноразово визнавався різними дослідниками найрозумнішим з «розумних» міст на планеті, причому його уряд йде ще далі і вважає, що працює над проектом «розумної нації» [5] (Smart Nation - назва програми міського розвитку Сінгапуру). Цілий ряд стартапів спільно створює рішення для Сінгапуру, які стосуються практично всіх сфер життя городян - від охорони правопорядку і автоматичної фіксації порушень до управління транспортною системою і енергоресурсами, водопостачання та охорони здоров'я. І це дає свої результати, наприклад, одна тільки система управління транспортними потоками здатна заощадити сінгапурським водіям десятки тисяч годин на рік.

Як вже було сказано вище, головним принципом роботи кібер-фізичних систем можна назвати глибоку взаємозв'язок між їхніми фізичними і обчислювальними елементами. «Мозок» системи у вигляді ІІ та інших технологій отримує дані від сенсорів в реальному світі, аналізує ці дані і використовує їх для подальшого управління фізичними елементами. Завдяки такій взаємодії кіберфізична система здатна ефективно працювати в умовах, що змінюються, як аналог людського організму або сучасна компанія, яка аналізує ситуацію на ринку, щоб розробити саме той продукт, який йому зараз потрібен. При цьом цикл керування - Отримання даних - обробка даних - керування при работе системи кожен раз повинен давати позитивні результати і створювати нову цінність.

За наявності кіберфізичної системи постає можливість створення та уніфікації великої кількості розподілених систем, що кінець-кінцем дозволяє впровадження інтернету. Наявність засобів узгодження внутрішніх та зовнішніх параметрів дозволяє підвищення надійності певних компонентів кіберфізичної системи та здатність виступати в якості самодостатнього функціонального модуля [3,4]

1.3 Постановка задачі узгодження зовнішніх та внутрішніх параметрів системи без конфліктуючих показників

Нехай надано систему, що описується певним вектором внутрішніх та зовнішніх показників:

$$□^{□} = \{□^{□}_{□}\}_{□=\underline{1,□}}, □^{□} = \{□^{□}_{□}\}_{□=\underline{1,□}}, □ = \{□_{□}\}_{\underline{1,□}} \quad (1.1)$$

де $□_{□}^{□}$- керований показник, $□_{□}^{□}$- некерований показник, $□_{□}$- зовнішній показник

Наведені показники обираються за певні випадкові величини, вимагатимемо від даних випадкових величин наявності детермінованої функції, такої, що:

$$∀□ = \underline{1,□}\ ∃□_{□}: □^{□} \to □, □: □^{□} \to □: □_{□}(□^{□}) + □_{□}(□^{□}) + □_{□} = □_{□}$$
$$ξ_{□} \sim □(0, □^2) \quad (1.2)$$

Існування таких функцій - достатньо загальна постановка задачі, для коректності наступних міркувань вважатимемо дані функції вимірними, а множину точок, що в них функція має розрив вимірну із мірою 0. Всі неперервні функції та кусково неперервно-диференційовні функції належать до такого класу. Вважатимемо, що користувач-проектувальник не має прямого доступу до заданих функціональних залежностей і тому не має можливості зформувати значення вихідних функцій на основі відомих показників.

Зазначимо, що у формулі (2) вимагається наявність адитивного зв'язку між компонентами, зазначимо те, що даний вид залежності між компонентами також відповідає припущенню незалежності керованих та некерованих показників. Це є достатньо природнімм припущенням, що відповідає відсутності надлишкової інформації щодо компонент вектора $□^{□}$, в противному випадку інформація такого рода сприяла би можливості встановлення керування над компонентами вектора $□^{□}$ за рахунок компонент вектора $□^{c}$.

Також вважатимемо компоненти вектору $□$ незалежними та неконфліктуючими в тому сенсі, що вони є додатньо корельованими на

довільному проміжку довжини більшої ніж задана наперед похибка ε вони є додатньо корельованими.

Використання функції експоненти дозволяє впровадити мультиплікативну залежність у випадку необхідності. Але ми припускатимемо незалежність даних векторів на даний момент.

Тепер задача узгодження зовнішніх та внутрішніх параметрів системи зводиться до наступної задачі:

$$\forall\varepsilon > 0 \forall \square_0 > 0 \exists \square > 0: ||\square^{\square}{}_1 - \square^{\square}{}_2||_{\square} \leq \square \rightarrow \forall \square = \underline{1, \square}$$
$$\square(||\square^{1}{}_{\square} - \square^{2}{}_{\square}||_{\square} \geq \square) \leq \square_0 \qquad (1.3)$$

де $\square^{\square}{}_1, \square^{\square}{}_2$ - відображають вектори контрольованих змінних

Таким чином постановка даної задачі в першій, найпростішій постановці полягає у відтворенні функції:

$$\delta(\square, \square^{\square}) \qquad (1.4)$$

вона відповідає допустимій контрольваній зміні вхідних параметрів, що забеспечує обмеження зміни вихідних параметрів.

Даний тип задачі буде розв’язуватися за умови відсутності інформації щодо чітких функціональних залежностей. Структурно постановка нагадує означення збіжності за ймовірністю, проте тут маємо дещо більш складні умови, про те розглядається лиш часткова збіжністю за параметрами, вважаючи, що інші залишаються нам невідомими. Для формалізації даної задачі розглядатиметься найгірший випадок, тому для усунення даної невизначеності будемо розглядати наступну постановку:

$$\forall \varepsilon > 0 \forall \square_0 > 0 \exists \square > 0: ||\square^{\square}_1 - \square^{\square}_2||_{\square} \leq \square \rightarrow \forall \square = \underline{1, \square}$$
$$\square(\square \square \square_{\square^{\square}} \quad ||\square^1_{\square} - \square^2_{\square}||_{\square} \geq \square) \leq \square_0$$

де $\square^{\square}_1$, $\square^{\square}_2$ - відображають вектори контрольованих змінних,

$\square^{\square}$ - відображає неконтрольований показник

вона відповідає найгіршій реалізації заданих вихідних величин-критеріїв за умов неконтрольваних показників.

Для розв'язку цієї задачі користуватимося елементами регресійного аналізу та теорії наближень. Оскільки справжні залежності між компонентами невідомі наперед то для розв'язку даної задачі можна скористатися пошуком наближення невідомих функцій. Ми розглянемо декілька підходів у наступному розділі.

1.4. Методи наближення невідомих функціональних залежностей

В даному розділі буде описано підхід до виконання моделювання

Регресійна модель є функція незалежної змінної і параметрів з доданою випадкової змінної. Параметри моделі налаштовуються таким чином, що модель найкращим чином наближає дані. Критерієм якості наближення (цільовою функцією) зазвичай є середньоквадратична помилка: сума квадратів різниці значень моделі і залежною змінною для всіх значень незалежної змінної як аргумент. Регресійний аналіз - розділ математичної статистики і машинного навчання. Передбачається, що залежна змінна є сума значень деякої моделі і випадкової величини. Класична постановка задачі регресії має наступний вигляд[5]:

$$\square = \square(\square) + \square$$

(1.5)

де $\square(\square)$ – *функція регресії,*

ε – *випадкова величина із нульовим середнім*

Даний вираз в точності співпадає із виразом (1.2). За наявності вибірки пар аргумент-значення, то використовуючи метод максимальної правдоподібності, можна оцінити функцію регресії на основі наступного вигляду[7]:

$$\square^*(\square) = \square\square\square\square\square\square_{\square\in\square}\square(\square|\square,\square,\square) \qquad (1.6)$$

Якість аппроксимації сильно залежить від вибору функціонального простору $\square$ а також від корельованості вхідних компонент X. Опишемо декілька можливих варіантів вибору функціональних просторів:

1. $\square$ - простір лінійних функцій, тоді розв'язком даної задачі оптимізації та умові некорельнованого нормального шуму ε із сталою дисперсією:

$$\square(\square,\square) = \sum_{\square=1}^{\square} \quad \square_{\square}\square_{\square} + \square. \qquad (1.7)$$

Розв'язок при такому функціональному просторі зводиться до задачі квадратичної оптимізації, що приймає наступний вигляд:

$$\square = (\widehat{\square}^{\square}\widehat{\square})^{-1}\widehat{\square}^{\square}\square = \square\square\square\square\square\square_{\square\in\square^{\square}}||\widehat{\square}\square - \square||,$$

де $\widehat{\square}$ - розширена матриця спостереження із одиничним стовпчиком для врахування вільного коефіцієнту.

2. Спеціальні типи лінійної регресії - поліноміальна регресія, експоненціальна регресія, вони відповідають факту вибору наступного класу функцій, а саме:

$$\square_{\square} = \square \circ \square,$$

де Φ: $\square^{\square} \rightarrow \square^{\square}$, $\square(\square) = (\square_1(\square), \cdots, \square_{\square}(\square))$

Даний вид регресії враховує використання покоординатного перетворення перш ніж застосувати лінійну регресію. Як вже зазначалося, простір перетворень може бути або недостатньо репрезентативним при дуже поганому виборі сімейства перетворень, або ж навпаки занадто репрезентативним, що створює загрозу перенавчання моделі.

3. Метод найближчих сусідів припускає те, що модель має таку властивість, що близькі за положенням аргументи мають відповідати близьким значенням вихідної величини, таку ідею використовує регресії найближчих сусідів, загальний вигляд моделі приймає вигляд[5-6]:

$$\square(\square, \square) = \frac{1}{\square} \sum_{\square \in \square_{\square}(\square)} \quad \square_{\square} \tag{1.8}$$

Недоліком зазначених вище моделей є велика схильність до перенавчання, якщо не використовувати певних штрафних чи обмежувальних методів. Моделі також є вразливими до викидів в спостережуваних даних, що призводить до необхідності попереднього проведення очищення або фільтрації. Метод найближчих сусідів також втрачає свою ефективність при збільшенні розмірності вхідного простору, оскільки велика кількість даних виявляється розміщеною на достатньо великій відстані та взаємовплив точок одна на одну стає дедалі меншим.

Таким чином враховується вплив лише перших к найближчих сусідів на значення вихідної змінної. Впровадження вагів, що пропорційні відстанням до точок-сусідів є також поширеною технікою, що відповідає ідеї того, що ближчі точки мають завдавати більший вплив на точку x:

$$\square(\square,\square) = \frac{1}{\sum_{\square\in\square_{\square}(\square)} \quad ||\square-\square_{\square}||} \sum_{\square\in\square_{\square}(\square)} \quad ||\square-\square_{\square}||\square_{\square} \tag{1.9}$$

Різні атрибути можуть мати різний діапазон представлених значень у вибірці (наприклад атрибут $\square_1$ представлений в діапазоні від 0.1 до 0.5, а атрибут $\square_2$ представлений в діапазоні від 1000 до 5000), то значення відстані можуть сильно залежати від атрибутів з великими діапазонами. Тому дані зазвичай мають бути спершу нормалізованими. Якщо певна ознака є більш важливою, можливим є збільшення вагів для певних компонент вектора $\square$.

Метод Групового Врахування Аргументів відповідає використовуванню ідеалогії рекурсивного перебору моделей, що мають відповідну структуру, а саме частіше за все обирають сімейство наближувальних функцій за поліноми Колмогорова-Габора[5,6]:

$$\square(\square) = \square_0 + \sum_{\square=1}^{\square} \quad \square_{\square}\square_{\square} + \sum_{\square=1}^{\square} \quad \sum_{\square>\square} \quad \square_{\square\square}\square_{\square}\square_{\square} + \cdots + \square_{123\ldots\square}\square_1\square_2\cdots\square_{\square} \tag{1.10}$$

Такий вибір обумовлений теоремою Стоуна-Вейерштраса, що зазначає, що довільна неперервна функція на компакті може бути наближена поліномом з наперед заданою точністю. Після використання певного ансамблю моделей, з них обираються найкращі за певним критерієм якості, наприклад, критерій $\square^2$ або середньоквадратична помилка. Як результат, отриманий ансамбль моделей

може бути побудований на основі або обрізаного поліному Колмогорова-Габора, або ж на основі повного набору. Після досягнення певного рівня складності, або ж при досягненні певного рівня якості наближення, процедура припиняється. Якщо ж процедура не припиняється, моделі отримані на попередньому кроці використовують на наступному кроці як вхідні аргументи. Дану модель можна вважати одним із типів нейронних мереж, її також називають поліноміальною нейронною мережею.

Інші методи, включають також можливість використання штучних нейронних мереж задля того, щоб наблизити невідомі функціональні залежності.

Проста модель штучного нейрона, що включає деякі властивості біологічних нейронів - це перцептрон, представлений Розенблатом у 1957 році. Більш конкретно, модель Перцептрона одного штучного нейрона включає кілька входів (рис. 1.2), ефекти яких визначаються змінними вагами, окремим виходом (який, однак, може бути скопійований/роздутий довільну кількість разів), інтеграцією вхідних сигналів та процесом регулювання ваг:

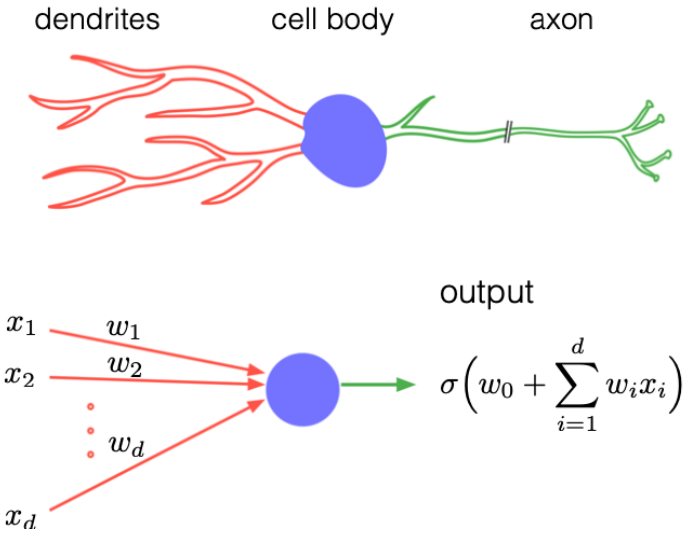

Рисунок 1.2 - Модель штучного нейрона в порівнянні з біоло гічним нейроном

У моделях з використанням ациклічних графіків нейрони можна безпосередньо розглядати як «одиниці обробки інформації», для яких напрямки країв визначають напрямки потоку інформації. Математично кажучи, у цьому випадку кожному нейрону призначається функція з простору із розмірністю, що відповідає кількості вхідних ребер, та образ якої відповідає кількості вихідних ребер. В принципі, це може бути будь-яка функція. На практиці, однак, традиційною структурою цієї функції є аффінне відображення, складене з неаффінною та покоординатно діючою функцією σ: R → R. Тут "функція активації" σ зазвичай фіксована так, що всі вільні параметри моделі містяться в афінних перетвореннях. Якщо у випадку рекурентних мереж доводиться мати справу з часовими рядами, то у випадку мереж з прямим зв'язком вся мережа тоді характеризується однією функцією від вхідного до вихідного простору. У пошаровому випадку це має такий вигляд:

$$\circ_{\square=1}^{\square} (\square \circ (\square_{\square} *. + \square_{\square}))\square$$

Перевагою штучних нейронних мереж постає можливість повного вивчення довільної скінченної множини точок із неперервними значеннями та дискретними значеннями. Дані результати можна довести принаймні із використанням щонапройстішої функції активації $\sigma(\square) = \square(\square > 0)$.

Даний метод спирається на універсальну теорему наближення, що стверджує для довільної функції, що є неперервною на компакті можна побудувати наближення у вигляді [8]:

$$\square = \sum_{\square=0}^{\square} \quad \square_{\square}\square(< \square_{\square}, \square > + \square_{\square}) + \square_{\square} \tag{1.11}$$

де $\sigma(\square)$ - довільна Ліпшиць нерервна функція, що прямує до 0 при $\square \to -\infty$, та до 1, при $\square \to \infty$. Зазвичай функції активацій нейронних мереж обираються за функцію сигмоїда, чи функцію:

$$\square\square\square\square(\square) = \square\square\square\{0, \square\} \tag{1.12}$$

Для такої функції активації також існує відповідна теорема, що дозволяє схоже наближення. Кількість доданків в даній функції відповідає кількості нейронів в одношаровій нейронній мережі. Таким чином вже одношарової нейронної мережів вистачає, щоб наблизити довільну неперервну функцію. Для того, щоб пояснити існування потужність нейронних мереж представлення, можна звернутися до Теореми Колмогорова про суперпозицію[5,6]:

$$\square(\square_1, \cdots, \square_{\square}) = \sum_{\square=0}^{2\square} \quad \square(\sum_{\square=1}^{\square} \quad \square_{\square}\square_{\square}(\square_{\square})) \tag{1.13}$$

В свій час дана теорема дозволила розв'язати одну з проблем Гільберта, що дозволило створити

Можна також довести, що необхідний розмір тренувальної вибірки залежить сильно від кількості параметрів в багатошаровій нейронній мережі, проте є доведені оцінки на мінімальну кількість параметрів, необхідну для потенційного представлення функції певної розмірності.

У 2000 роках було доведено, що збільшення кількості шарів є більш суттєвим для покращення наближувальних властивостей нейронної мережі, це пояснюється можливістю зведення складних функцій до композиції простіших, які в свою чергу можуть бути описані як одношарові нейронні мережі. Таким чином багатошаровий перцептрон використовує ідею, що перехід від шару до шару відповідає збільшенню рівня композиції функцій. Фактично цей факт можна пояснити наступним емпіричним спостереженням, що кількість осциляцій, що нейрона мережа здатна наблизити змінюється адитивно із зростанням кількості нейронів у шарі та мультиплікативно із зростанням кількості шарів. Це ,звичайно, покращує наближувальні характеристики моделі (рис. 1.3).

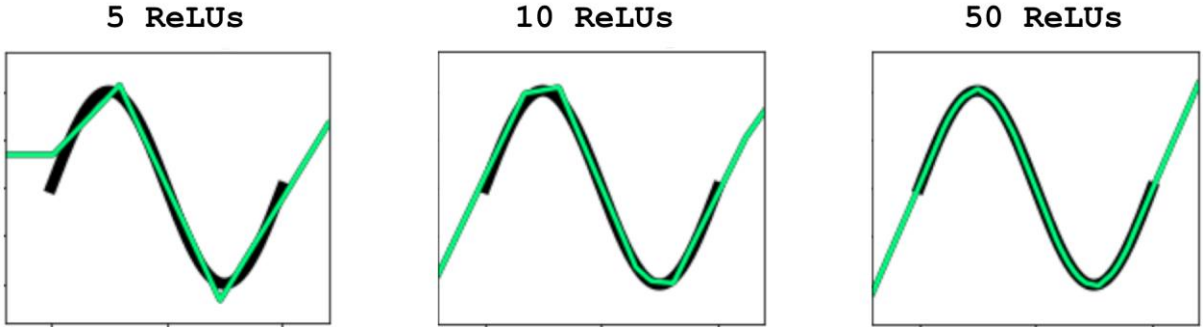

Рисунок 1.3 - Приклад послідовності наближень функції нейронною мережею

Ще одним важливим методом наближення зазначених у попередньому розділі функціональних залежностей є методом опорних векторів, що може бути описаний наступним чином[6]:

$$\min_{\square} 0.5 * ||\square|| + \square \sum_{\square=0}^{\square} |\square_{\square}|$$

$$\{|\square_{\square} - < \square, \square_{\square} > | \leq |\square|_{\square} + \square \qquad (1.14)$$

Така постановка задачі вимагає пошуку такого вектору □, що всі точки знаходяться в певному епсілон-околі від прямої, що має цей вектор за нормаль. Тут параметр C відповідає тому, яку увагу ми приділяємо врахуванню елементів, що не вкладаються в окіл шириною *2*□ з центром на прямій. SVR - це потужний алгоритм, який дозволяє нам вибирати, наскільки ми толерантні до помилок, як через прийнятну похибку помилки (є), так і шляхом налаштування нашої толерантності до виходу за межі цієї допустимої частоти помилок (Рис. 1.4). Великим плюсом даної моделі є можливість віднайти її розв'язок за допомогою засобів опуклої оптимізації. Наприклад, введення дуальної задачі до даної робить проблему розв'язуваною аналітично, що дозволяє достатньо швидко отримати результати моделювання.

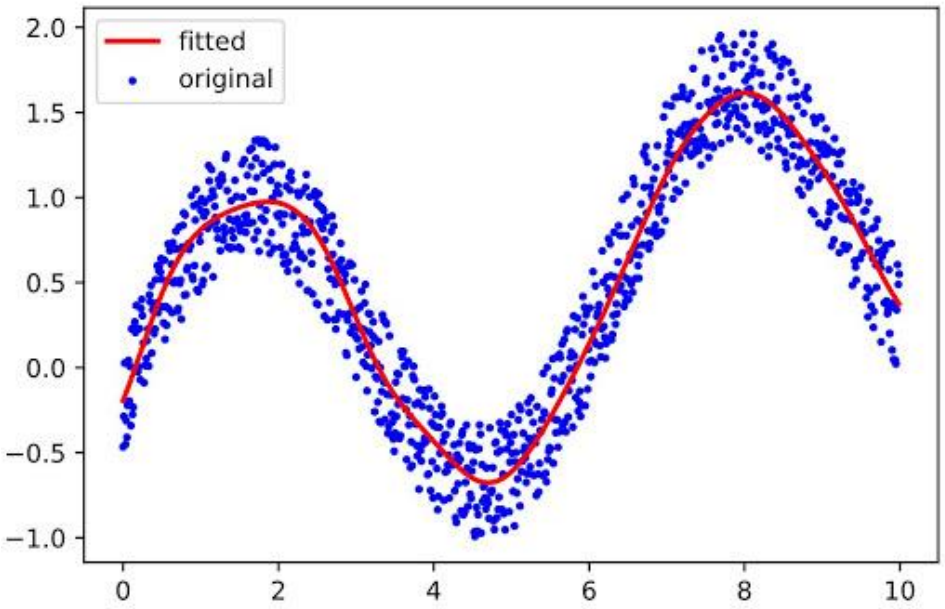

Рисунок 1.4 - Приклад моделі опорних векторів

Часто застосування стандартної евклідової метрики певного порядку виявляється недостатнім для репрезентативності даних, тоді на допомогу приходять так званий ядерний метод. Його застосування буде описано у наступному параграфі.

Нарешті розглянемо моделі, що використовують кусково сталі функції для апроксимації невідомої функціональної залежності. Дані моделі відповідають розбиттю множини визначення функції на певні області, що утворюють покриття та кожна компонента якого не перетинається з іншими, або ж формально, як його ще називають - дерево прийняття рішень (рис. 1.5)[10]:

$$□(□) = \sum_{□=1}^{□} □(□ \in \times_{□=1}^{□} [□_{□□}, □_{□□}]) \quad (1.15)$$

Таке розбиття виявляється непоганим наближуванням при використанні ієрархічного моделювання, де за основу беруться зважені комбінації сукупності більш простих моделей. Дана ідея має назву градієнтного бустингу.[10]

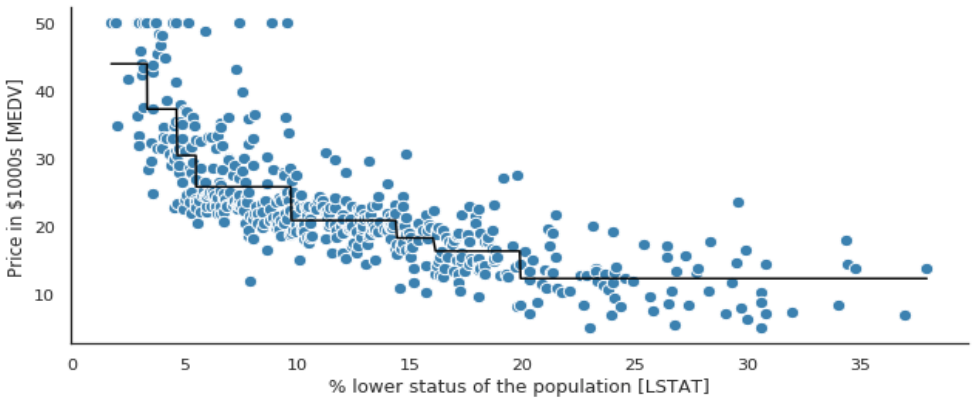

Рисунок 1.5 - Приклад моделі дерева прийняття рішень

Таким чином в даному параграфі було надано короткий екскурс в інструментарій, що використовуватиметься для попереднього відновлення функціональних залежностей.

1.5 Методи покращення якості побудованих моделей

В даному розділі будуть розглянуті певні методи для підвищення якості описаних у попередньому розділі моделей.

Для попередження перенавчання моделі часто необхідно розглянути постановку задачі наближення тренувальних даних у наступному вигляді[10]:

$$\square^* = \square\square\square\square\square\square_{\square\in\square}\square(\square|\square,\square,\square) + \square(\square) \qquad (1.16)$$

де $\Omega(\square)-$ штрафна функція, що залежить виключно від функції, що вибирається із заданої сім'ї функцій.

Введення такої штрафної функція часто є природнім, при зміні підходу до вибору функції регресії, а саме Метод Максимального Апостеріорного розподілу, що відповідає[6]:

$$\square = \square\square\square\square\square\square_{\square\in\square}\square(\square|\square,\square,\square) \qquad (1.17)$$

Такий підхід відповідає пошуку моди апостеріорного розподілу та є точковою оцінкою невідомих параметрів. При такій постановці задачі, звичайна лінійна регресія за умови, що її коефіцієнти мають нормальний розподіл із заданою дисперсію відповідає так званій $\square_2$регуляризації, що описується штрафним виразом вигляду[6]:

$$\Omega(\square) = 0.5 * \square * ||\square||_2,$$

де -дисперсія апріорного розподілу параметрів

Така регресія спонукає модель шукати параметри, що мають приблизно однаковий масштаб. Вибір розподілу Лапласа відповідає впровадженню □$_1$регуляризації, що має наступну функцію штрафа[9,10]:

$$\Omega(\square) = \square * ||\square||_1$$

Така регресія спонукає вектор параметрів мати якомога меншу кількість ненульових компонент, що відповідає розрідженим векторам. Така процедура штрафує модель за надлишкову складність, примушуючи її шукати компроміс між необхідністю використання певного параметру та його внеском в пояснення спостережуваних даних. Варто згадування внесення розподілу дисперсії за рахунок оберненого гамма розподілу з метою моделювання робастної регресії - такої, що вона є стійкою до викидів та відповідає кінець-кінцем розподілу Стьюдента.

Для тренування нейронних мереж вкрай важливим виявляється стандартизація та нормалізація вхідних даних на кожен шар нейронної мережі, оскільки це стабілізує навчання та попереджує перенавчання. Кожен рівень нейронної мережі має входи з відповідним розподілом, на який під час навчального процесу впливає випадковість у ініціалізації параметра та випадковість у вхідних даних. Вплив цих джерел випадковості на розподіл вхідних даних та на внутрішні шари під час навчання описується як внутрішній зсув коваріації. Хоча чітке визначення, насправді, відсутнє, явище, яке спостерігається в експериментах,-це зміна засобів та відхилень вхідних даних для внутрішніх шарів під час навчання.

Також важливою є техніка так званого відкидання зайвих нейронних зв’язків таким чином, метод регуляризації штучних нейронних мереж, призначений для зменшення перенавчання мережі за рахунок запобігання складних коадаптаціі окремих нейронів на тренувальних даних під час навчання. Термін «dropout» (вибивання, викидання) характеризує виключення

певного відсотка (наприклад 30%) випадкових нейронів (що знаходяться як в прихованих, так і видимих шарах) на різних ітераціях (епохах) під час навчання нейронної мережі. Це дуже ефективний спосіб усереднення моделей всередині нейронної мережі. Цей метод також належить до класу так званих ансамблевих методів.

Наведемо 2 відомих та дієвих методи - Ансамблювання та Градієнтний бустинг, перший метод намагається це мета-алгоритм ансамблювання машинного навчання, призначений для підвищення стабільності та точності алгоритмів машинного навчання, що використовуються у статистичній класифікації та регресії. Це також зменшує відхилення та допомагає уникнути надмірного комплектування. Хоча він зазвичай застосовується до методів дерева рішень, його можна використовувати з будь-яким типом методів. Ансамблювання - це окремий випадок підходу моделювання усереднення. При цьому прості регресори, наприклад, як вже, зазначалося, дерева, звичайні лінійні регресії або навіть неглибокі нейронні мережі піддаються усередненню з метою зменшення дисперсії вихідних випадкових величин.

Градієнтний бустінг - це метод машинного навчання для регресії, класифікації та інших завдань, який виробляє модель прогнозування у вигляді ансамблю слабких моделей прогнозування, зазвичай дерев рішень. Коли дерево рішень навчається, отриманий алгоритм називається деревами з посиленням градієнта, який зазвичай дає кращі результати ніж просто усереднені моделі. Модель будується поетапно, де чергова модель намагається виправити помилки попередніх при наближенні. Доведено, що така процедура гарантовано дає кращі результати.

Зазначимо наприкінці найбільш поширений метод попередження перенавчання на контролю складності моделі. Він полягає в застосуванні так званої перехресної валідації, ідея якої полягає в розбитті тренувальної вибірки на частини, одна з яких використовується в якості перевірочної, а інші в якості

тренувальної. Як результат, гарна модель має демонструвати приблизно однакові результати на кожній з відкладених частин.

1.6 Постановка задачі відшукання множини Парето для заданої системи із вхідними та вихідними показниками

При проектуванні складного технічного виробу найважливішою задачею є раціональний вибір параметрів і узгодження вимог до зовнішніх (області значення) і внутрішніх (області визначення) показників виробу при апріорно відомих обмеженнях на показники зовнішнього впливу.

Розглянемо можливості розв'язання даної задачі на прикладі раціонального погодженого формування множини Парето за вихідними даними виробу.

В даній роботі буде також розглянуто встановлення множини (або ж принаймні її частини) Парето у наступному сенсі, якщо маємо задану систему на кшталт тієї, що її було задано у параграфі 1.3, то вважатимемо, що вихідні показники конфліктують, якщо кореляція між ними є меншою від певного від'ємного значення. Таким чином при збільшенні, або зменшенні одного показника, інший є більш схильним до зростання або спадання. Пошук множини Парето будемо шукати наступним чином[9]:

$$\{\square \in \square^{\square} | \forall \square \in \square^{\square}: \square(\square) \geq \square(\square) \textit{ та принаймні одна нерівність є строгою}\}$$

Із означення множини Парето випливає те, що всі її елементи є непорівнюваними між собою. Зазначимо, що у випадку неконфліктуючих показників (визначення такої множини критеріїв надавалося у параграфі 1.3),

проблема пошуку множини Парето зводиться до розв’язання задачі оптимізації, що має наступний вигляд:

$$□ = □□□□□□_{□}\{□□□_{□}\{□_{□}(□)\}\} \quad (1.18)$$

Питання локального пошуку для багатовимірних вхідних параметрів будемо розглядати розбиття області визначення на прямокутні області, що утворюють покриття. Цікавим є динаміка кореляції на кожному прямокутнику

Якщо вихідні величини є неконфліктуючими глобально (на кожному проміжку). Випадок на певних проміжках відбувається зміна кореляції із позитивної на негативну чи навпаки, то є сенс в пошуку максимуму для проміжків взаємного зростання та мінімуму на кожному із проміжків спадання.

Таким чином проміжки проміжки додатньої кореляції дозволяють локальне спрощення задачі.

Щодо проміжків від’ємної кореляції, пошук компромісу полягає в накладанні ваг на кожен із критеріїв та оптимізації зваженої функції. Випадок зміни залежності можна трактувати з такої самої точки зору.

Одже за умови пошуку багатовимірної множини Парето, окрему увагу привертає можливість раціонального вибору ваг для оптимізації зваженої вихідної функцій вхідних параметрів. Дана процедура проводиться звичайно після побудови та тренування моделі, що описує вхідні дані на основі дискретних підвибірок. Процедуру вибора ваг буде описано у наступному розділі. Вона будується на основі нейроних мереж з використанням механізму “Attention”.

1.7 Висновки першого розділу

Отже в першому розділі було детально описано постановку задачі пошуку множини Парето за наявності декількох конфліктуючих та неконфліктуючих показників та багатьох вхідних показників, що є внутрішніми спостереженнями та описують вихідні показники. Такі показники бувають двох типів, а саме, по-перше контрольовані та неконтрольовані. Тоді із постановки задачі виявляється доцільним розбиття задачі на дві підзадачі, перша з яких відповідає відновленню невідомої функціональної залежності між вхідними та вихідними показниками, а друга безпосередньо відповідає формуванню множини Парето.

Для розв’язку першої задачі виявляється доцільним застосування методів машинного навчання, що допомагають віднайти невідому функціональну залежність у бажаній формі та у вигляді конкретних зв'язків. До таких моделей належать передусім, моделі класичного регресійного аналізу, моделі нейронних мереж, ансамблеві моделі та багато інших. Нарешті друга задача торкається безпосередньо необхідної для нас мети формування певної множини, що допомагає віднайти необхідну задачу пошуку множини Парето, використовуючи методику зважения із врахуванням потенційно відсутньої експертної оцінки, щодо того, як можна порівняти між собою багато критеріїв за важливістю.

Таким чином описані постановки задач можуть бути застосовані для формування допустимого діапазону зміни вхідних показників, маючи за мету гарантувати обмежену згори зміну вихідних показників. В наступному розділі буде детально описано методи для розв’язку даної задачі, що пропонуються в цій роботі.

РОЗДІЛ 2: МЕТОДИ УЗГОДЖЕННЯ ОБЛАСТЕЙ ВИЗНАЧЕННЯ-ЗНАЧЕНЬ

2.1 Загальна характеристика методів узгодження областей визначення-значень

Оскільки за основу мету даного параграфа ставиться категоризація методів узгодження областей визначення-значень, то доцільним можна вважати розпочати із того, що поділити запропоновані методи на певні категорії і в подальшому описати застосування кожного із методів в контексті коректності ситуації, в якій його можна застосувати.

Запропоновані методи можна поділити на методи з обмеженням помилки та без обмеження, методи, що використовують неперервність та не використовують, методи, що включають розв’язок задач математичного програмування, та не включають. Можливими також є методи Монте-Карло.[11]

До першої категорії методів, що включають методи з обмеженням помилки будемо відносити методи, що роблять гіпотезу про мажорантність швидкості зміни функції в залежності від зміни аргументу, так обмеження, що накладатимемо в тому випадку, коли використовуватимемо такі методи має наступний вигляд:

$$\forall \varepsilon > 0 \exists \square > 0 : \square < \square\square\square\square(\square, \square) \qquad (2.1)$$

де $\square\square\square\square(\square, \square)$ – *поліном* $\square$ *степені від* $\square$.

Таке обмеження дозволяє принаймні обмежити зміну згори певним виглядом поліному, та не намагатися відшукати занадто великий інтервал для зміни аргументу. Категорію методів, що не включають обмеження буде розглянуто трохи пізніше із використанням методів зведення таких проблем до проблем попереднього типу.

Методи, що використовують неперервність функції відповідають використанню моделей, що побудовані на використанні моделей неперервних функцій, такі методи передусім спираються на аппарат моделювання описаний детально в попередньому параграфі, оскільки довільні наближувальні функції, що їх було запропоновано у попередньому параграфі мають властивість неперервності. Такі підходи ґрунтуються на тому факті, що при апроксимації достатньо гарних функцій можливим є відшукання локальної константи Ліпшиця для кожної точки, що вона належить області визначення функції і як результат можливим стає наближення з використанням пропорційності помилки наближення функції до похибки аргументу наступним чином:

$$|\square(\square) - \square(\square)| \leq \square|\square - \square| \qquad (2.2)$$

Методи, що на противагу не припускають неперервності функції мають за мету використання розбиття множини визначення на об’єднання підмножин, на кожній з яких функція є неперервною та застосування підходу із константою Ліпшиця на кожній із описаних підобластей.

Методи, що спираються на використання математичного програмування, а саме дискретного програмування, базуються на попредньому зведені задачі, що її було поставлено у попередньому параграфі до задачі пошуку найближчої точки, що порушує відстань до ключової функції на необхідну наперед задану

величину ε. Така задача змішаного програмування (лінійного та бінарного) гарантує віднайдення точки, яку варто обрати за межу допустимої області. Важливим нюансом в застосуванні такого методу є той факт, що розв'язок задач мішаного програмування є NP-повною задачею, а тому небезпечним є складні функціональні залежності, оскільки час розв'язку таких задач для кожної точки не може бути обмежений поліномом від розміру вхідних даних.

Методи, класифікацію яких було наведено у даному параграфі можуть бути імплементовані із використанням насамперед штучних нейронних мереж, що є вкрай важливим, оскільки це є одним із найперспективніших напрямів розвитку методів штучного інтелекту, нижче наведено схематичну класифікацію даних методів (рис. 2.1):

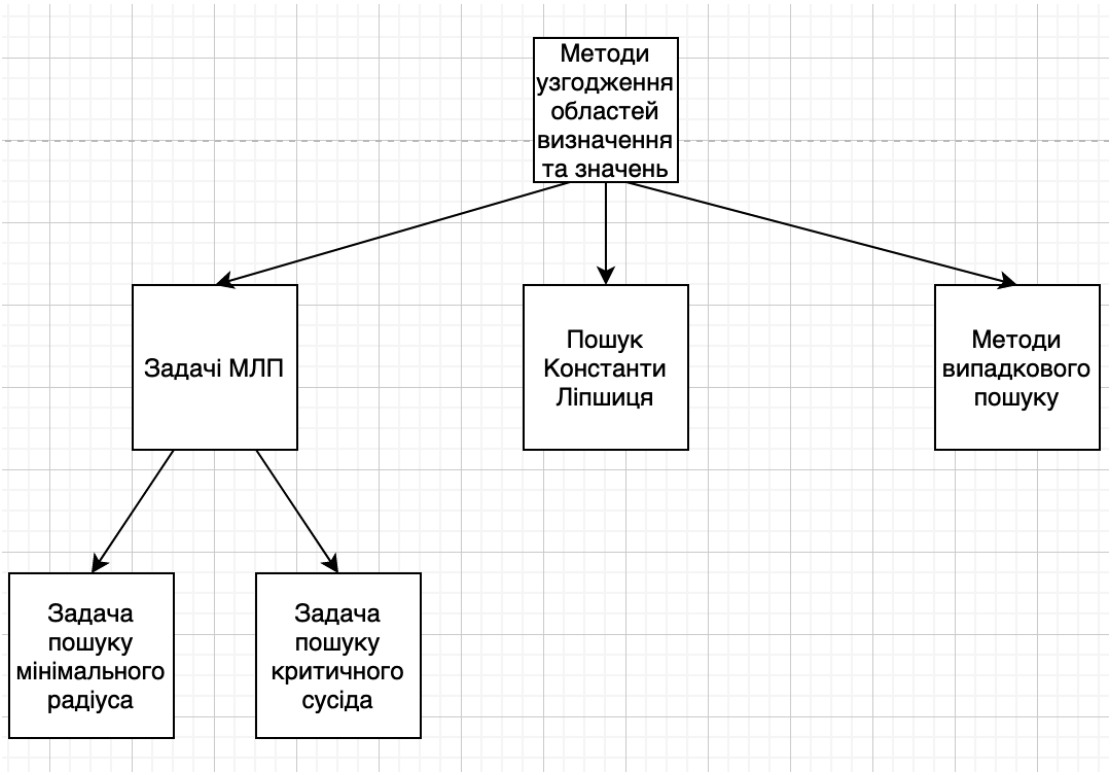

Рисунок 2.1 - Класифікація методів узгодження областей визначення та значень

В наступних розділах ми детально опишемо реалізацію кожного із запропонованих методів, їх переваги та недоліки буде виокремлено після того, як спершу кожний з цих методів буде описано.

2.2 Методи Мішаного Лінійного Програмування (МЛП)

Розглядаються методи мішаного лінійного програмування, що можуть бути застосовані для пошуку критичних точок, що порушують відхилення функції від значення у точці-центрі більше ніж на певне задане наперед значення ε. Перший метод, що буде розглянуто саме і передбачає пошук найближчої точки, в якій різниця за модулем між дійсним значенням функції та у даній точці більше від заданого числа ε, оскільки більшість сучасних нейроних мереж будуються із використанням таких функцій активацій як ReLU(x) чи LeakyReLU(x), а обидві з цих функцій є кусково лінійними, то виявляється перспективним формулюваня наступної задачі математичного програмування :

$$\min_{\square^1,\ldots,\square^{\square},\square^1,\ldots,\square^{\square}\square} \{||\square - \square||_2^{\,2}\ \}$$

за умови що:

$$\square^{(\square+1)} = \square^{\square}\square^{\square} + \square^{\square}\ \forall\square\ \in \underline{1, \square - 1} \tag{2.3}$$

$$\square^{(\square)} = \square\square\square\square(\square^{(\square)})\ \forall\square\ \in \underline{2, \square} \tag{2.4}$$

$$|\square^{\square} - \square(\square^*)| \leq \square \tag{2.5}$$

$$\square^1 = \square^1$$

Тут невідомими є всі проміжні змінні (векторні) при прямому поширені в нейронній мережі.

В такій постановці задачі можна помітити, що маємо задачу нелінійного програмування, якщо в якості норми вибрати стандартну евклідову норму на площині. Зазначимо, що тут використовуються два види обмежень, що ускладнюють дану задачу в замкненому вигляді, а саме обмеження (2.4) та (2.5), покажемо, що якщо виконати певні еквівалентні перетворення, то дану задачу можна звести до задачі змішаного квадратичного програмування, а у певних, гарних випадках і до задач звичайного квадратичного програмування. Почнемо із того, що перетворимо обмеження (2.4) наступним чином:

$$\square = \square\square\square\square(\square) \leftrightarrow \{\square \leq \square - (1 - \square)\square, \square \leq \square\square, \square \geq 0, \square \geq \square, \square \in \{0,1\}\} \quad (2.6)$$

де $\square, \square$ - відповідно апріорно відомі нижні обмеження на вхідну величину x

Можна перевірити, що дана система нерівностей дійсно коректно описує обмеження рівності для функції $\square = \square\square\square\square(\square)$. Отже введення такої системи нерівностей дозволяє позбутися нелінійного обмеження. Зазначимо, що апріорні константи $\square, \square$ - обираються з наступних міркувань, або ж достатньо малими та великими, або ж уточнюються під час навчання нейронної мережі перед тим, як проводити розв'язок відповідної задачі математичного програмування. Важливість та перевагу другого підхода над першим проілюструємо проблемою розв'язків задач дискретного програмування. Тут при, $\square > 0$ маємо те, що обмеження є лінійним, а при $\square \leq 0$ - і зовсім рівне 0, тому якщо $\square > \square \geq 0$ або $\square < \square \leq 0$ то відсутня необхідність в застосуванні методів дискретного програмування. Проблема виникає лише у третьому випадку, коли $\square < 0 < \square$ і постає необхідність використовувати апарат дискретного програмування. Саме тому вибір екстремальних обмежень, що не є достатньо обмежуючими відносно вхідної змінної можуть призвести до експоненціального вибуху під час розв'язку даної задачі[11].

На практиці також зустрічаються ще й інші види функцій активацій, що є кусково лінійними, продемонструємо схожу функцію активації[10,12]:

$$□□□□6(□) = □□□\{□□□\{0, □\}, 6\} \quad (2.7)$$

Дана функція активації є важливою на теренах сучасних архітектур, а саме відомої MobileNetv3, отже продемонструємо можливість представлення цієї функції у вигляді системи з лінійно-дискретних обмежень:

$$□ = □□□□6(□) \leftrightarrow \{□ \leq □ - (1 - □_1)□, \}$$
$$□ \geq □, □ \leq □□_1□ \geq 6□_2, □ \leq 6□_2, □_2 \leq □_1, □_□ \in \{0,1\}$$

Те саме стосується довільних кусково-лінійних функцій. Розглянемо тепер можливість перетворення обмеження (2.5) на сукупність лінійних обмежень, що остаточно дозволять нам перетворити задачу на задачу мішаного квадратичного програмування. Отже обмеження (2.5) подамо у наступній формі:

$$□ \; □ - □ \; (□) \leq □^+_□$$
$$-□ \; □ + □ \; (□) \leq □^-_□$$
$$\varepsilon^+ \; + □^- \; \leq □$$
$$\varepsilon^+ \; \geq 0, □^- \; \geq 0$$

Ідея даного представлення полягає в розбитті на 2 частини нерівності, перша відповідає за ситуацію, в якій відхилення відходить в одну сторону, та навпаки. Отже, нарешті, задача може бути представлена у наступному вигляді:

$$\min_{□^1,\ldots,□^□, □^1,\ldots,□^□□} \{||□ - □||_2^{\,2} \; \} \quad (2.6)$$

за умови що:

$$□^{(□+1)} = □^{(□)}□^{(□)} + □^{(□)} \; ∀□ \in \underline{1, □ - 1}$$

$$□^{(□)} \leq □^{(□)} - (1 - □^{(□)})□^{(□)}, □ \leq □^{(□)}□^{(□)}, □^{(□)} \geq 0, □^{(□)} \geq □^{(□)}, □^{(□)} \in \{0,1\}^{□} \; ∀□ \in \underline{2, □}$$

$$|□^{□} - □(□^{*})| \leq □$$

$$□^{1} = □$$

Тут -розмірність векторів $□^{(□)}$, $□^{(□)}$

Таким чином маємо наявну задачу змішаного квадратичного програмування. Оптимальним розв'язком цієї задачі є ,відповідно, найближча точка до точки, окіл якої досліджується, що відстань від значення в цій точці до значення в знайденій точці має абсолютну різницю, що не перебільшує ε.

Розглянемо можливість пошуку іншої задачі в термінах задачі мішаного лінійного програмування. За основну мету ставиться обернена задача, а саме пошук найбільшого відхилення функції при заданому околі значення аргумента:

$$\min_{□^{1},\ldots,□^{□},□^{1},\ldots,□^{□}□} \{|□(□^{*}) - □|\} \qquad (2.7)$$

за умови що:

$$□^{(□+1)} = □^{□}□^{□} + □^{□} \; ∀□ \in \underline{1, □ - 1}$$

$$□^{(□)} = □□□□(□^{(□)}) \; ∀□ \in \underline{2, □}$$

$$(2.5) \; ||□ - □^{*}||^{2}_{2} \leq □$$

$$□^{1} = □^{*}$$

Тут наперед відомим є окіл точки і вимагається знайти найбільше відхилення в цьому околі. Деякі обмеження можуть бути переформульовані так само як і ті, що розглядалися в контексті попередньої задачі. Залишається остаточно звести дану задачу до задачі змішаного лінійного програмування шляхом того, що ввести додаткову змінну[11]:

$$\square = \square(\square(\square^*) - \square) + (1 - \square)(\square - \square(\square^*))$$
$$\square \in \{0,1\}$$

Таким чином з урахуванням попередніх еквівалентних перетворень, маємо остаточно:

$$\min_{\square^1,\ldots,\square^\square,\square^1,\ldots,\square^\square \square} \{|\square(\square^*) - \square|\} \quad (2.7)$$

за умови що:

$$\square^{(\square+1)} = \square^{(\square)}\square^{(\square)} + \square^{(\square)} \; \forall \square \in \underline{1, \square - 1}$$
$$\square^{(\square)} \leq \square^{(\square)} - (1 - \square^{(\square)})\square^{(\square)}, \square \leq \square^{(\square)}\square^{(\square)}, \square^{(\square)} \geq 0, \square^{(\square)} \geq \square^{(\square)}, \square^{(\square)} \in \{0,1\}^\square \; \forall \square \in \underline{2, \square}$$
$$\square = \square(\square(\square^*) - \square) + (1 - \square)(\square - \square(\square^*))$$
$$\square \in \{0,1\}$$
$$\square^1 = \square$$

Отже в даному розділі було розглянуто можливість представлення задачі узгодження областей визначення та значень у вигляді задач лінійного змішаного програмування. Два можливі підходи дозволяють віднайти залежність околу точки, за відомим обмеженням околу значення функції та навпаки, маючи відоме значення околу точки, віднайти допустиму область

зміни значення. Обмеження даного підходу полягають передусім в тому, що архітектура нейронної мережі є фіксованою. Якщо навчання проводилося без належної підготовки, то розв'язок таких задач є позбутнім сенсу. Також структурна обмеженість активаціями типу □□□□(□), □□□□□ □□□□(□), □□□□6(□), що є кусково лінійними. В більшості випадків таке обмеження може бути не важливим для формування такої задачі, оскільки використання таких функцій вважається кращою практикою на сьогоднішній день.[11]

Наприкінці зазначимо, що для розв'язку подібних задач часто застосовують метод опуклої релаксації, де дискретні обмеження типу □ ∈ {*0,1*} замінюються на обмеження типу □ ∈ [*0,1*]. Як результат така задача дозволяє швидке та ефективне оцінювання верхньої чи нижньої межі на справжнє значення в точці оптимуму.[8]

2.3 Методи пошуку константи Ліпшиця

Важливим методом для обмеження зміни функції за відомої зміни аргументу та навпаки є використання константи Ліпшиця, аргументація такого підходу побудована на теоремі Лагранжа, що дозволяє омбеження зростання функції за відомої зміни аргументу наступним чином[13,14]:

$$\forall \square, \square\ \exists \square \in [\square, \square] || \square(\square) - \square(\square)|| = ||\square\square(\square)||\ ||\square - \square|| \qquad (2.8)$$

$$\text{тут під } [\square, \square] = \{\square | \exists \square \in [0,1]: \square = \square\square + (1 - \square)\square\}$$

Отже базовою ідеєю може стати пошук максимума норми матриці Якобі відображення, що використовується для моделювання. Таким чином перший метод спирається на точну оптимізацію норми матриці Якобі.

Оскільки функцію задано табличним наближенням, то після підстановки моделі, першим кроком є вибір моделі, оскільки з огляду на теорему Лагранжа, функція має бути принаймні неперервно диференційовною на інтервалі між обраними точками, то обирати варто моделі, що гарантують таке наближення. Прикладами таких моделей є нейронні мережі, моделі регресії та багато інших. Моделі, що представлені ансамблями дерев є, очевидно, недиференційовними відображеннями і як результат не можуть бути застосовані для даного підходу.

Якщо модель задовільняє накладеним вище обмеженням, виявляється можливою чисельна оптимізація, наприклад методом субградієнтного спуску. Важливим є той факт, що сучасні моделі нейроних мереж часто спираються на використання активацій типу □□□□(□), □□□□□ □□□□(□), □□□□*6*(□), що є кусково лінійними. Відповідне наближення є також кусково лінійним. Тому необхідне наближення необхіднно або згладити в місцях розриву похідних. Іншим рішенням може бути обмеження застосування моделей такими, що використовують лише нескінченно диференційовні функції.

Другий метод полягає в обмежені константи Ліпшиця, спираючись на той факт, що більшість функцій активації, таких як сигмоїдальна чи функція гіперболічного тангенсу мають ці величину за одиницю, тому залишається лише проаналізувати поведінку нейронної мережі, як композиції лінійних слоїв та функцій активації, розташованих у певному порядку, можна спостерігати, що[13,14]:

$$\square_{\square} = 1, \square \in$$
$$\{\square\square\square\square\square\square\square(\square), \square\square\square\square(\square), \square\square\square\square(\square), \square\square\square\square\ \square\square\square\square(\square)\}$$
$$\square_{\square\square+\square} = ||\square|| \quad = \underset{||\square||=1}{\square\square\square}||\square\square|| = \square_{1} \tag{2.9}$$

Вираз (2.9) на пряму пов’язаний із результатами аналізу, він носить спеціальну назву - перше сингулярне значення. Таким чином, остаточно константа Ліпшиця мережі може бути оцінена наступним виразом:

$$\square_{\square\square} \leq \prod_{\square=1}^{\square} \quad ||\square||_{\square} \tag{2.10}$$

Більш складні нейроні мережі, що включають, наприклад згорткові чи рекурентні також можуть бути застосовані та для них можна знайти відповідне обмеження константи Ліпшиця.

2.4 Методи Випадкового пошуку

Нарешті, розглянемо останний тип методів, що представляє для нас інтерес, методи випадкового пошуку. Такі методи побудовані на створені наступної архітектури нейронної мережі (рис. 2.2):

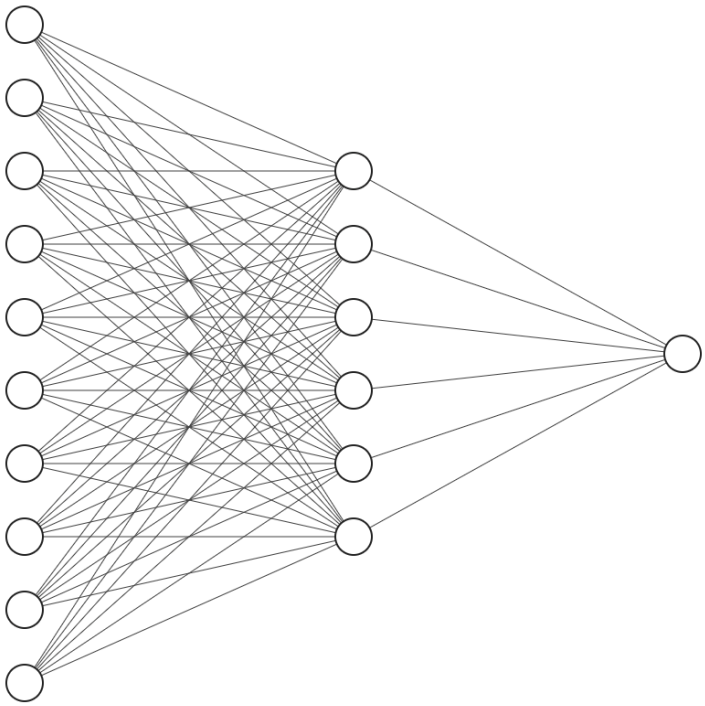

Рисунок 2.2 - Архітектура мережі

Дана мережа навчається таким чином, що намагається охопити зміну функції в кожній точці області визначення наступним чином, а саме:

1. На вхід подається точка x з області визначення.
2. Мережа видає ширину інтервалу.
3. Із отриманого інтервалу із центром в точці x.
4. Мережа, що за допомогою неї було отримано наближення вихідної множини функцій-критеріїв пропускає через себе отримані під час семплювання точки та визначає, чи знаходяться вони в допустимому околі чи ні.
5. Таким чином оцінюється якість першої мережі, та відбувається її навчання.

Зазначимо по-перше, що функція втрат для такої нейронної мережі має бути відповідною, а саме зважена функція двох компонент:

$$\square_1 \square_1(\square) + (1 - \square_1)\square_2(\square)$$

Тут перша функція має штрафувати мережу за невірні прогнози але й також штрафувати за занадто малу довжину інтервалу. Оптимізація лише першої компоненти відповідатиме довжині інтервалу навколо точки x 0, а другої - нескінченність. Звідси маємо задачу багатокритеріальної оптимізації.

Зазначимо, що під час побудови архітектури, що відповідає даній задачі в гру вступає велика кількість гіперпараметрів, що визначають складність нейронної мережі. Тому для гарної роботи алгоритму важливим рішенням може бути проведення оптимізації гіперпараметрів.[15,16,17]

Конкретний вибір ваг та функцій втрат буде описано у наступному параграфі.

2.5 Порівняння запропонованих методів, їх переваги та недоліки

Отже в даному розділі було розглянуто 3 різних підходи до узгодження областей визначення та значень заданих функцій. Основною метою даного розділу було продемонструвати потенціал відповідної теорії у застосуванні до аналізу функцій однієї змінної, вихідна структура яких є невідомою, а наближення будується на основі певного сімейства наближувальних функцій. На завершення наведемо порівняння описаних вище методів, їх переваги та недоліки.

Методи, що передбачають розв’язок задач мішаного лінійного програмування належать до класу NP-повних задач, тому за певних обставин методи їх розв’язку можуть бути достатньо тривалими і тому неприйнятними для рішення задач в умовах обмеженого часу. Опукла релаксація може

допомогти, проте однозначно втрачається якість наближення. До їх недоліків також можна віднести структурні обмеження, що накладаються на сімейство наближувальних функцій. Це мусять бути нейроні мережі із певним типом функцій активації, а саме - кусково лінійними. Перевага цього методу полягає в можливості точного встановлення найближчої точки, яку варто обрати за межу інтервалу.

Підходи, що вони засновані на використанні сталої Ліпшиця є достатньо простими методами, їх можна використовувати під час навчання нейронної мережі або іншого алгоритма, щоб слідкувати за динамікою ширини довжини інтервалу. Їх недоліком є складність обчислення при зміні структури наближувальних функцій. До того ж пошук сталої Ліпшиця може бути проблематичним для функція із сильними осцеляціями. Передусім такий прорахунок може бути виправлений, якщо розглянути методи локальної константи Ліпшиця.

Нарешті, методи випадкового пошуку є універсальними в тому сенсі, що вони застосовують нейронні мережі, як апроксимацію відповідної довжини інтервалу, що дозволяє отримати швидкий аналіз чутливості заданої функції, якщо заданим є тільки точка, та було проведено відповідне попереднє навчання. Даний метод вимагає на жаль коректного вибору ваг та функція втрат для того, щоб гарантувати збіжність та знайдення раціонального компромісу.

Таким чином всі з описаних методів можуть бути застосовані в залежності від задачі, яка розв’язується. Конкретне порівняння даних методів буде проведено у наступних розділах під час розв’язку задачі пошуку множини Парето для конфліктуючих критеріїв.

2.6 Висновки другого розділу

Отже, в другому розділі було запропоновано низку методів, що можуть бути успішно застосовані для розв’язку задач узгодження областей визначення та значення за умови попередньо невідомих функціональних залежностей. Всі із запропонованих методів мають різну природу та передбачають застосування різноманітних методів, що належать до категорії системного аналізу та аналізу даних. Було окреслено сукупність переваг та недоліків цих методів. Під час розробки та аналізу даних методів враховувалися сучасні тенденції на ринку розроблення програмних продуктів, створення програмного забезпечення, було ідентифіковано перспективні напрямки та траєкторії дослідження.

В наступному розділі буде детально описано підходи до пошуку множини Парето для заданої системи вхідних та вихідних параметрів. Проведено порівняльний аналіз отриманих моделей.

РОЗДІЛ 3: МЕТОДИ ФОРМУВАННЯ МНОЖИНИ ПАРЕТО ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Опис моделі формуваня Множини Парето

У попередньому розділі було детально викладено методи встановлення залежностей між змінами вихідних та вхідних функцій, в даному розділі буде наведено результати формування множини Парето, описано підходи, які можуть бути застосовані. Описано моделі та їх результати. Також буде проведено аналіз отриманих результатів, проведено порівняльний аналіз отриманих результатів. За основний метод імплементації описаних підходів, будемо використовувати мову Python та фреймворк Pytorch. Отже, опишемо передусім архітектуру та ідеологію нейронної мережі, що її буде застосовано для формування множини Парето. Для цього поступово описуватимемо процес формування, починаючи від попередньої обробки даних, визначення функції витрат, метрик та ,нарешті, архітектури нейронної мережі, що може бути використана для встановлення множини Парето. Однієї із ключових позицій, що її притримуватимемося під час моделювання - трактування із позиції зображень та використання так званих моделей сегментації. Відновлене зображення буде після цього пропущене через згладжування із використанням методів, описаних у попередньому розділі.

3.2 Попередня обробка даних

На вхід нейронної мережі подається вектор вхідних параметрів, що має розмір 10000x10, таким чином обмежимося виробами із кількістю вхідних параметрів меншою або рівною за 10. Наведемо спершу приклади множини Парето, що є типовими та зустрічаються на практиці. Для цього використаємо двовимірний набір вхідних змінних та два критерії, оптимізація яких буде проводитися. Далі підрахуємо одним із класичних методів область Парето, використовуючи, наприклад, звичайним методом перебору (рис. 3.1). Зазначимо, що для апріорно невідомих функціональних залежностей це не виявляється можливим, до того ж експоненціально швидко зростає кількість елементів дискретизації, що використовуються при формуванні множини Парето(Додаток 1):

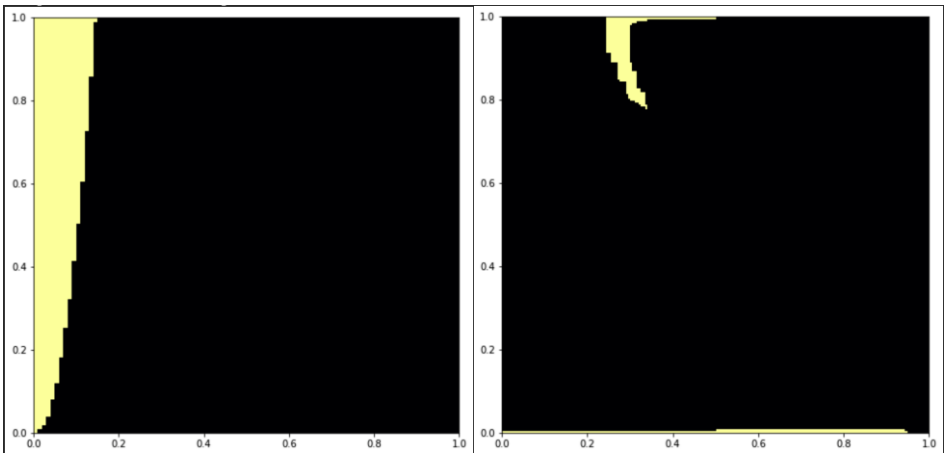

Рисунок 3.1 - Приклади множина Парето для тестових критеріїв $\{□_1^2 + □_2^2, -□_1^3 - □_2\}$ та $\{□_1^2 + □_2\ , -□_1\ - □_2^2\}$ із множиною вхідних параметрів $[0,1]^2$

Для попередньої обробки даних спершу проведемо нормування вхідних величин до відрізків $[0,1]$, це дозволить покращити тренування та звести відповідні критерії до однакового масштабу:

$$□_{□□} = \frac{□ - □□□(□)}{□□□(□) - □□□(□)} \quad (3.1)$$

Тепер для заданого набору даних, дискретизації, та значеннях функції на кожній із точок будуємо карту значень кожного критерію. На вхід нейронної мережі буде подаватися зображення, що має розмірність (кількість вихідних величин, розмір всіх векторів x).

3.3 Визначення функції витрат та метрик

Оскільки, задача, яку ми виконуємо побудована загалом на коректному співставленні областей, можна використати метрики, що дозволяють оцінити якість формування множини з позиції теорії міри, так, наприклад, індекс Жакарда:

$$□□□(□, \widehat{□}) = \frac{□(□ \cap \widehat{□})}{□(□ \cup \widehat{□})}, \quad (3.2)$$

де $□_{□}$, $\widehat{□_{□}}$ - відповідно справжня множина Парето, та зформоваа

За ідеального співпадіння, метрика дорівнює одиниці, вона також штрафує занадто жадібну класифікацію множини Парето, так визначення множини Парето за всю множину призведе до сталої помилки. Дану метрику буде обрано з позиції використання в моделі сегментації.

Індекс Жакарда нажаль не є диференційовним, того не може бути використаний в якості метрики. Його диференційовний аналог має наступний вигляд:

$$□_{□□□}(□, \widehat{□}) = \frac{\sum_{□=1}^{□} □_{□}\widehat{□_{□}}}{\sum_{□=1}^{□} □_{□}+\widehat{□_{□}}-□_{□}\widehat{□_{□}}}, \quad (3.3)$$

де -кількість критеріїв, $□_{□}$, $\widehat{□_{□}}$ - відповідно справжня множина Парето, та зформована

Дана метрика досягає оптимуму за співпадіння областей, проте має достатньо великі за нормою градієнти. Використання згляджування, дозволяє створити передумови для використання такої метрики:

$$□_{□□□}(□, \widehat{□})_{□} = □(1 - \frac{□+\sum_{□=1}^{□} □_{□}\widehat{□_{□}}}{□+\sum_{□=1}^{□} □_{□}+\widehat{□_{□}}-□_{□}\widehat{□_{□}}}) \quad (3.4)$$

де -кількість критеріїв, $□_{□}$, $\widehat{□_{□}}$ - відповідно справжня множина Парето, та зформована, α - коефіцієнт згладжування

Така функція витрат зберігає оптимальна значення, проте зменшує значення градієнту, що прискорює навчання та дозволяє запобігти ефекту вибухання градієнту, що призводять до неякісного навчання нейронної мережі. Таким чином визначилися із функцією втрат та метрикою.

3.4 Визначення архітектури нейронної мережі.

Обираємо архітектуру U-Net, що спирається на використання наступних важливих блоків, так, наприклад, важливим є згортковий блок, що має наступний вигляд (рис. 3.2):

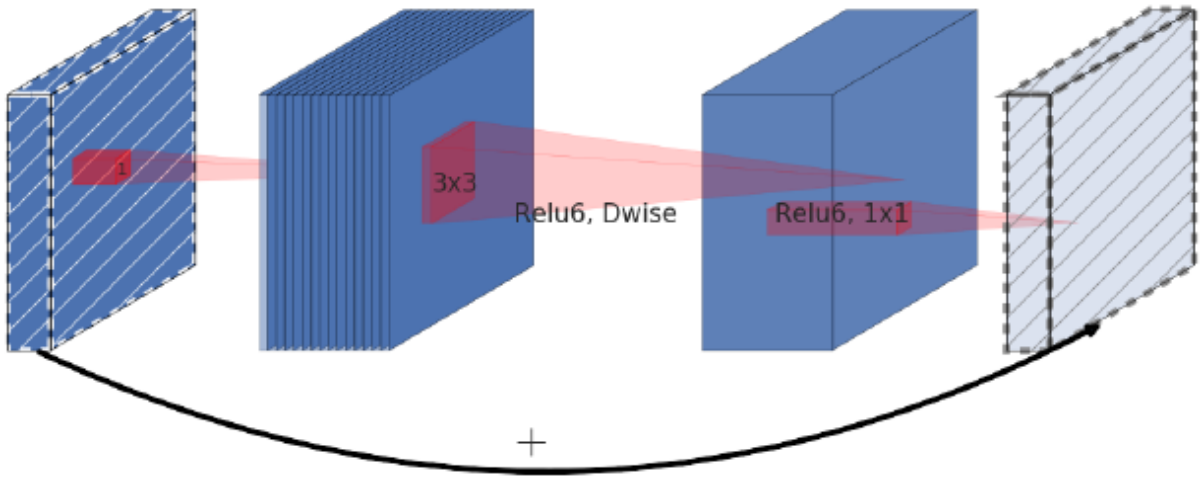

Рис. 3.2 - Згорткова нейронна мережа

Існує низка передумов появи згорткових нейронних мереж, а саме:

- багато нейронів в зоровій корі мають невелике локальне рецепторне поле (local receptive field), а тому реагують на зорові подразники;
- деякі нейрони реагують тільки на зображення горизонтальних ліній, в той час як інші - на лінії в різних напрямках;
- деякі нейрони мають більші рецепторні поля і реагують на більш складні образи, які є об’єднанням низькорівневих образів;
- висунули гіпотезу, що кожний нейрон пов’язаний тільки з декількома нейронами попереднього рівня.

Такі особливості побудови людського сприйняття інформації зумовили експертів в області Computer Science спромогтися спроектувати дані особливості на застосування сучасних нейронних мереж. Першою з таких моделей стала Neocognitron від К.Фукусіма у 1979-1980 роках. Основною одиницею згортковий нейронної мережі є слой згортки. Шар згортки включає в себе для кожного каналу свій фільтр, ядро згортки якого обробляє попередній шар за фрагментами (підсумовуючи результати поелементного добутку для кожного фрагмента). Вагові коефіцієнти ядра згортки (невеликі матриці) невідомі і встановлюються в процесі навчання.

Особливістю згортковий шару є порівняно невелика кількість параметрів, яке встановлюється при навчанні. Так наприклад, якщо вихідне зображення має розмірність 100 × 100 пікселів по трьом каналам (це значить 30000 вхідних нейронів), а згортковий шар використовує фільтри с ядром 3x3 пікселя з виходом на 6 каналів, тоді в процесі навчання визначається тільки 9 ваг ядра, однак по всім сполученням каналів, тобто 9 × 3 × 6 = 162, в такому випадку даний шар вимагає знаходження тільки 162 параметрів, що істотно менше кількості шуканих параметрів повно нейронної мережі. Також можна виокремити наступні властивості:

- зберігає структуру входу, а саме - порядок в одновимірному випадку, взаємне розташування пікселів в двовимірному випадку тощо, оскільки застосовується до кожної ділянки вхідних даних окремо;
- має властивість розрідженості, оскільки значення кожного нейрону шару залежить лише від невеликої долі вхідних нейронів. В повнозв’язній мережі, наприклад, кожний нейрон залежить від усіх нейронів попереднього шару;
- багаторазово використовує одні й ті ж ваги, оскільки вони повторно застосовуються до різних ділянок вхідного зображення.

Спершу вхідне зображення покривається невеликими вікнами, що дозволяють отримати ознаки на кожному такому вікні невеликою нейронною мережею. Таких згорток формується ансамбль, що в результаті застосування призводить до виникнення сукупності карти ознак, що після цього складаються у єдиний тензор із розміром каналів, що відповідає бажаній кількості ознак.

Результати цієї нейронної мережі знову представляється у вигляді зображення, замінюючи результат застосунку вікна на кожне рецепторне поле сумою чи іншою агрегацією. При цьому паралельно виконується декілька згорток, тим самим породжуючи лінійні активацій, які після чого подаються на

вхід на нелінійні активації (так звана детекторна стадія). Нарешті після отримання ознак, необхідно виконати операцію субдискретизації, в якості якої часто обирають пулінг.

Ідея згорткової мережі полягає в тому, що наявність тієї чи іншої ознаки є набагато більш важливою за позицію цієї ознаки. Тому, фактично, така мережа втрачає частину інформації, щодо позиції ознак, проте навчається їх наявності, тим самим досягається економія пам’яті. Задача ж пулінгу - забезпечення інваріантності щодо афінних перетворень.

Можливі варіанти пулінгу охоплюють взяття максимума по прямокутному вікну (МаксПулінг), середнього, застосування певних норм, таких як норми типу Lp.

Крок дискретизації вибирають рівним розміру вікна. Вхідна матриця ділиться на вікна, які не перетинаються, у кожному вікні вибирається невеликих трансформацій зображення, таких як зсув.

Якщо посунути зображення на певний відступ, то отримаємо, що значення результату макс пулінгу зміняться лише на половину, оскільки така операція чутлива лише до застосування операції максимуму.

Отже, в згорткових нейронних мережах виділяють два види операцій -

1. Витяг ознак (Feature Extraction) - Використання операції згортки із метою витягнення ознак, де положення не є важливим - більш важливим постає відносне положення (локальне) відносно інших ознак.
2. Відображення ознак (Feature Mapping) - Складання отриманих ознак у єдиний тензор - карту ознак.

Під час навчання мережа вчиться виокремлювати окремі фільтри для того, щоб отримати найкраще представлення у вигляді карти ознак. Для цього використовуються різні фільтри.

На противагу згортковим шарам ставитемо так звані антизгорткові шари. Їх головна ідея полягає в використанні операції, що, фактично є оберненою до операції згортки та відповідає збільшенню роздільної здатності зображень. Для цього між пікселями вхідного зображення вставляються пікселі, що мають певну залежність від ядра згортки та дозволяють збільшити її роздільну здатність (рис. 3.3).

Рисунок 3.3 - Операція антизгортки

Таким чином, одночасне використання операції згортки та антизгортки в ідеалі може призвести до відновлня зображення. Це стає в нагоді під час конструювання мереж, що в мають в основі процедуру стискання інформації (зображення) та отримання низькорозмірних характеристик, що дозволяють класифікацію чи відновлення. Такі архітектури базуються на послідовних/паралельних застосуваннях процедур згортки та антихгортки із метою відновлення вихідного зображення. В контексті нашої задачі, вхідним зображенням є саме багатоканальне зображення, елементами якого є значення

ключових функцій-критеріїв, що утворюють систему, для якої віднаходиться множина Парето. Такі архітектури мають назву автокодувальники, в загалом автокодувальник має наступний вигляд (рис. 3.4):

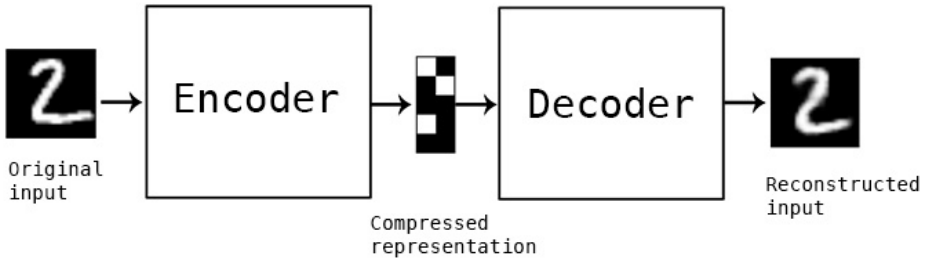

Рисунок 3.4. - Загальна структура автокодувальника

При цьому можна вважати, що при використанні такої архітектури відбувається неявне виокремлення ознак, що мають низьку розмірність, або, іншими словами, стиск інформації. Через це автокодувальник можна віднести до свого роду моделі навчання без вчителя. Ос інкільки за відсутності конкретного набору ключових ознак, що прогнозуються в процесі навчання, можна все ж таки провести виокремлення ознак за допомогою стиснення.

Тепер, коли всі ключові компоненти архітектури U-Net було охоплено, можна перейти до описання власне моделі, що буде використовуватися при формуванні множини Парето. Мережа U-Net вивчає відображення зображення із використанням послідності вкладених стискань та розширень. При цьому інформація отримана після стискання приєднується до інформації, отриманої після стискання, що призводить до того, що за умови, що стискання призводить до погіршення формування, модель матиме властивість відразу відкинути неякісну долю зображення, що її буде збережено у другій половині каналів (рис. 3.5).

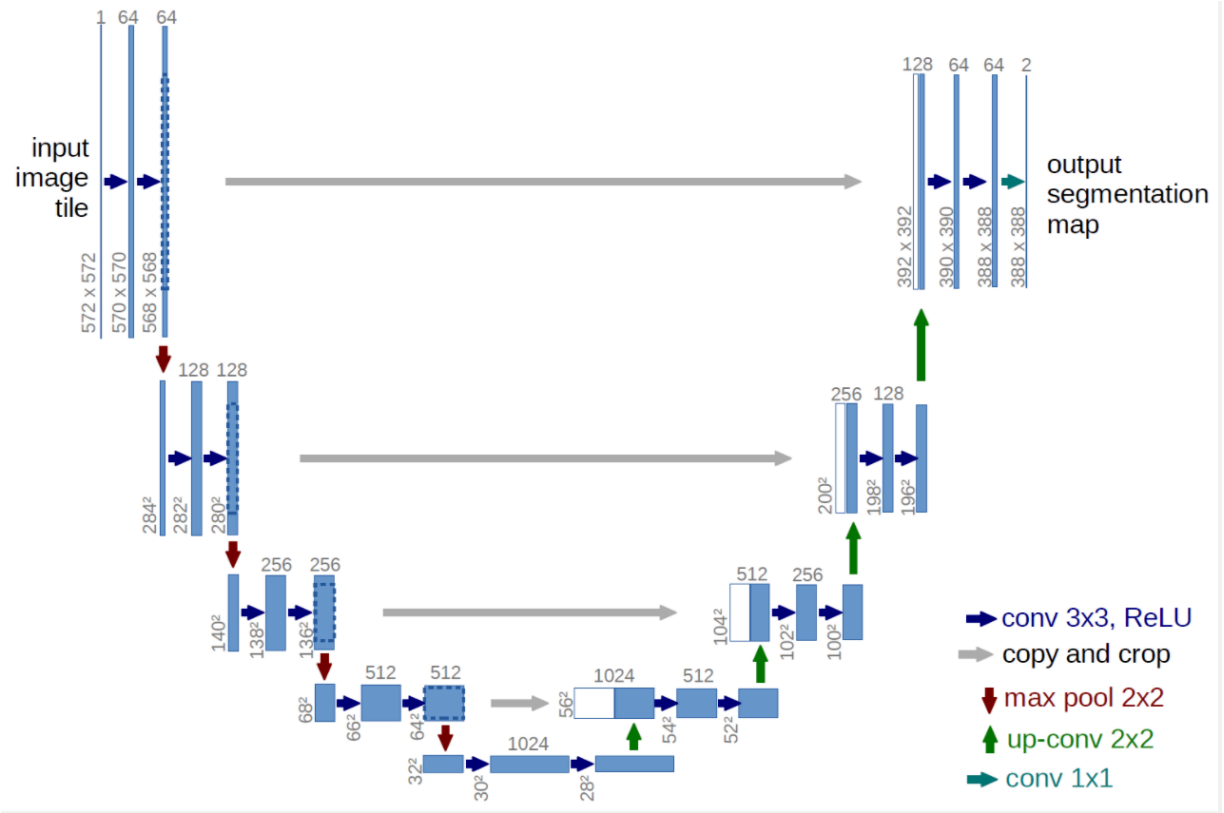

Рисунок 3.5 - Загальна архітектура мережі U-Net

Таким чином отримана вихідна сегментаційна карта може бути використана для встановлення факту приналежності частини певної долі області до множини Парето. Отже в даному підпункту було детально викладено ще один елемент запропонованої архітектури. В наступному розділі буде описано загальну модель відновлення множини Парето на основі описаних у другому розділі підходів та із використанням описаної в даному пункті сегментаційної моделі.

3.5 Загальний опис архітектури мережі

В даному розділі буде описано загальну архітектуру запропонованої мережі, що буде використовуватися для формування множини Парето (рис. 3.6):

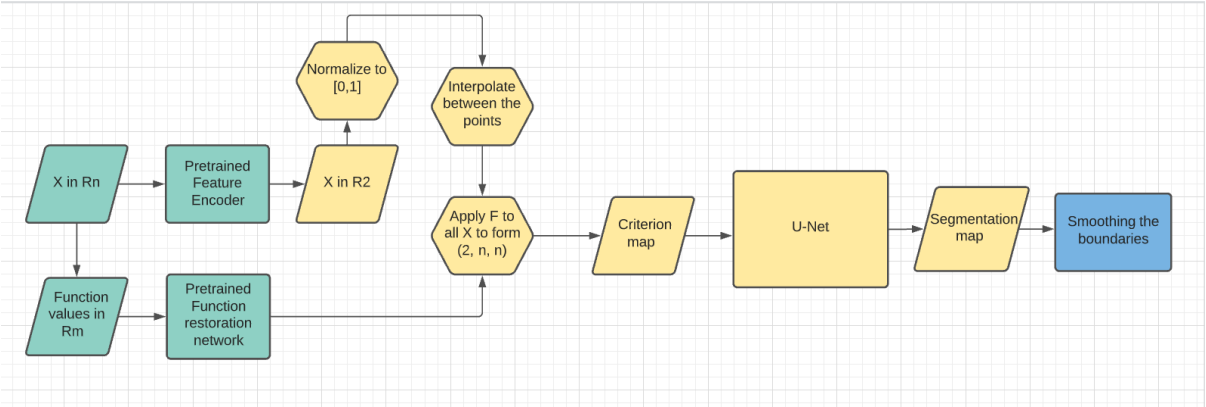

Рис 3.6 - Загальна структура формування множини Парето

Дана архітектура потребує певного пояснення. Варто розпочати із того, що для навчання відповідної сегментаційної моделі, необхідним постає навчання моделей, що мають, передусім, відновлювати функціональні залежності за наданими вибірками (загальний підхід було описано у розділі 1). Після відновлення відповідних функціональних залежностей, помітемо, що для використання підходу на основі сегментаційних моделей, зручним є можливість представлення даних у вигляді двовимірного зображення. Що накладає певні обмеження на архітектуру. Так, для збереження інформації, що наявна у даних, виконаємо зменшення розмірності із використання автокодувальника для вхідних даних. Це означає, що разом із відновленням функціональних залежностей необхідно також провести попереднє навчання автокодувальника для стиснення високовимірних ознак до двовимірних. Зазначимо, що можливість використання тривимірної згортки також може використана, проте в даній роботі таких підхід не буде обговорюватися.

Архітектура автокодувальника дозволяє проведення ефективної інтерполяції у просторі меншої розмірності, що і використовується в даній роботі для отримання щільної множини у просторі аргументів.

Після того, як отримано достатньо велику кількість зображень, можна застосувати до отриманої множини відновленні функціональні залежності та отримати кількість зображень, що відповідає кількості вихідних параметрів. Конкатенація таких зображень в одне велике зображення вздовж першого виміру призводить до формування багатокального зображення.

Отриманий таким чином набір даних пропускається для формування множини Парето через мережу типу U-Net, архітектуру якої було описано у попередньому пункті. Остаточно проводиться згладжування отриманих множин із використанням технік, описаних у другому розділі. Для наданих функціональних залежностей формується модель, що відновлює чутливість функції відносно зміни аргументів. Після цього отримана апроксимація розширюється на краях із застосуванням приєднання колової області на межі області Парето випадковим чином. Така фільтрація гарантовано зберігає слабку зміну значень аргументів за наданої зміни відновлених функціональних залежностей.

3.6 Тренування нейронних мереж

В даному розділі буде описано процес тренування нейронної мережі, описаної у попередньому пункті. Як вже зазначалося, структура включає в себе чотири моделі, що узгоджуються в єдиний механізм. До них належать автокодувальник для проектування вхідних параметрів великої розмірності, нейронна мережа для відновлення функціональних залежностей за табличним

даними, сегментаційна модель U-Net та модель для відновлення чутливості функцій (узгодження областей значень та визначень). Всі ці моделі, за виключенням автокодувальника для вхідних параметрів вже описувалися в даній роботі.

Тренування автокодувальника зводиться до використання наданих користувачем векторів внутрішніх параметрів системи із метою пошуку двовимірного представлення системи вхідних параметрів. При цьому в якості функції помилок застосовується звичайна середньоквадратична похибка, метрика співпадає із функцією витрат. Після тренування залишаємо виключно кодувальник, декодувальник не використовуватимемо.

Тренування моделі для відновлення функціональних залежностей проводитимемо звичайним чином, а саме на вхід подається вектор закодованих вхідних величин для відновлення функціональних залежностей. Таку процедуру можна трактувати як пошук композиції відображень, перше з яких представляє з себе відновлення точки із двовимірного простору, а друга безпосереднє наближення невідомих функціональних залежностей.

Для фільтрації можна обрати довільний метод, описаний в попередньому розділі, наприклад, метод встановлення константи Ліпшиця, чи метод мішаного лінійного програмування.

Нарешті, опишемо тренування мережі, типу U-Net, навчання якої має бути проведеним напередодні до навчання інших моделей. Навчання цієї моделі базуватиметься на використанні набору даних, що представляє з себе сукупність різних видів поліномів, та вручну підраховану для них область Парето. Основна проблема полягає в формуванні датасету. Кожне зображення вимагає достатньо тривалого формування. Так, в середньому обчислення області Парето для достатньо гарної дискретизації із кроком 0.01 на одиничному відрізку потребує приблизно 3-4 секунд, кількістю критеріїв при цьому не грає ролі. При формуванні тренувальної та перевірочної вибірки, важливим є покриття всіх можливих випадків та перебір без повторень, а також

без співпадіння критеріїв (в цьому випадку задача втрачає ,фактично, сенс, оскільки зводиться до максимізації). Спираючись на теорему Стоуна-Вейерштраса, за щільну множину в просторі неперервних функцій можна обрати множину поліномів. Оскільки критерії є еквівалентними, треба гарантувати симетрію. Так, розбиваємо для кожного степеня многочлена множину значень вільного члена на області, що не перетинаються, це гарантуватиме неспівпадіння функцій для різних критеріїв. Для цього казатимемо, що функції у Для обмеження простору функцій обиратимемо лише многочлени, що мають обмежений носій коефіцієнтів та обмежені одним степенем. Для боротьби з еквівалентними критеріями обмежитися варто лише поліномами із цілими, взаємопростими коефіцієнтами, що обмежені певної константою згори.

Таким чином в даному підрозділі було описано процедуру тренування та побудови описаної у попередньому підпункту архітектури. Не варто забувати про те, що обов’язковою є перевірка кожної складової архітектури на перевірочному наборі даних перед тим як продовжувати переходити до наступного етапу. В наступному розділі зробимо висновки стосовно запропонованої архітектури, її складових та іншого.

3.7 Висновки третього розділу

Отже в даному розділі було детально описано всі необхідні компоненти архітектури для формування області Парето для заданої системи функціональних залежностей наданих у табличній формі для довільної кількості вихідних та вхідних параметрів.

Першим кроком стало описати дані та форму в якій їх варто подавати на вхід модель сегментації, виокремлено обмеження, що дозволило, кінець-кінцем, вирішити в якій формі, та які компоненти варто внести до моделі для повноцінної реалізації архітектури.

Другим кроком було визначення головних складових моделі, що включають в себе автокодувальник для вхідних характеристик системи, модель для відновлення невідомих функціональних залежностей, моделі для узгодження областей значень та визначень системи внутрішніх та зовнішніх показників та сегментаційної моделі для зображень множини Парето.

Окремою задачею стало детально описати процес навчання кожної компоненти, так перші дві моделі навчаються за класичними схемами мінімізації середньоквадратичної помилки. Модель сегментації на основі архітектури U-Net було вирішено налаштувати з використанням метрики функції втрат Жакарда, що спонукає модель відновлювати невідомі функціональні залежності так, щоб гарантувати щонакрайще співпадіння областей в сенсі міри площі.

В наступному параграфі буде описано отримані результати, наведено криві навчання, наведено результуючі зображення відновлених множин Парето для різної системи критеріїв.

РОЗДІЛ 4: РЕЗУЛЬТАТИ ЗАСТОСУВАННЯ МОДЕЛІ

4.1 Завдання параграфу

Поведінка описаної моделі була досліджена на прикладі декількох прикладів, із застосуванням різних видів фільтрації. Серед яких варіант із застосуванням константи Ліпшиця, використанням методів мішаного лінійного програмування та методу випадкового пошуку. Процес навчання було попередньо описано в кінці третього розділу. Цей розділ ставить за мету порівняти отримані результати та дійти висновку, щодо найкращого використання моделі. Для імплементації використовувалася мова програмування Python та пакет Pytorch, для візуалізації використовувався сервіс Neptune.ai. Для навчання використовувалися потужності kaggle. Для оптимізації гіперпараметрів використовували Optuna.

4.2 Результати тренування моделей

Тренування допоміжних моделей, а саме автоенкодеру для внутрішніх показників системи, а також моделі для відновлення функціональних залежностей було проведено із використанням 3000 записів із кількістю внутрішніх показників 10 та зовнішніх показників 3. (див. рис. 4.1-4.2)

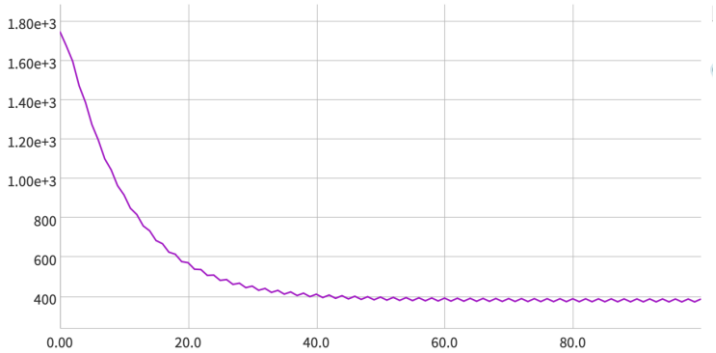

Рисунок 4.1 - Графік навчання моделі автокодувальника із lr=0.05025 (валідація)

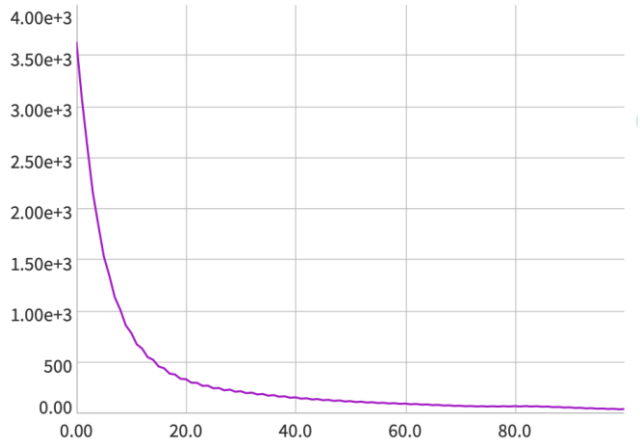

Рисунок 4.2 - Графік моделі відновлення із lr=0.00174 (валідація)

Обидві моделі досягли достатньо непоганих результатів. Навчання моделі формування множини Парето описано нижче. Для цього, за методом описаним у попередньому розділі, було сформовано вибірку із 1000 тестових критеріїв та внутрішніх параметрів, що будуть застосовуватися для навчання моделі. Логування результатів навчаня проводитемо із застосуванням фреймворку tensorboard, оскільки Neptune.ai є погано інтегрованим із фреймворком PytorchLightning, за допомогою якого виконувалося моделювання. (рис. 4.3)

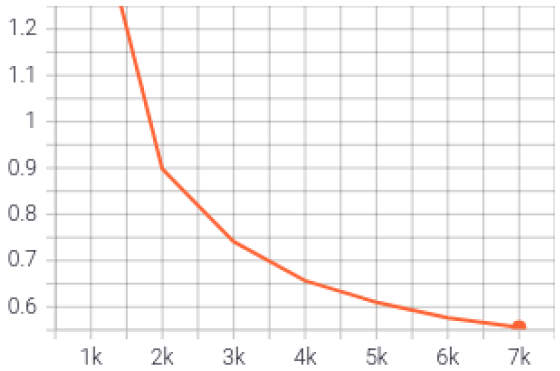

Рисунок 4.3 - Графік навчання моделі сегментації із lr=0.0001 (валідація)

За 9 годин модель виконала 7 тисяч епох та досягла достатньо гарної якості сегментації області Парето за наданим вхідних ознак, отриманих із застосуванням попередніх двох моделей. Таким чином було проведено тренування всіх моделей, знайдено оптимальний крок навчання та оптимальний розмір мініпакету, що застосовувався під час навчання. В наступному підрозділі буде продемонстровано формування множини Парето та порівняння із дійсними результатами спершу для відомого набору функцій без застосування фільтру для оброблення граничних значень.

4.3 Аналіз отриманих результатів

Розглянемо сукупність критеріїв, виду:

$$□_1^2 + □_2^2 -> □□□, □_1^2 - 6□_1 - □_2^2 + 2□_2 + 2 -> □□□ \quad (4.1)$$

де $□_1, □_2 \in [0,4]$

Дійсну область Парето зображену знизу (рис. 4.4)

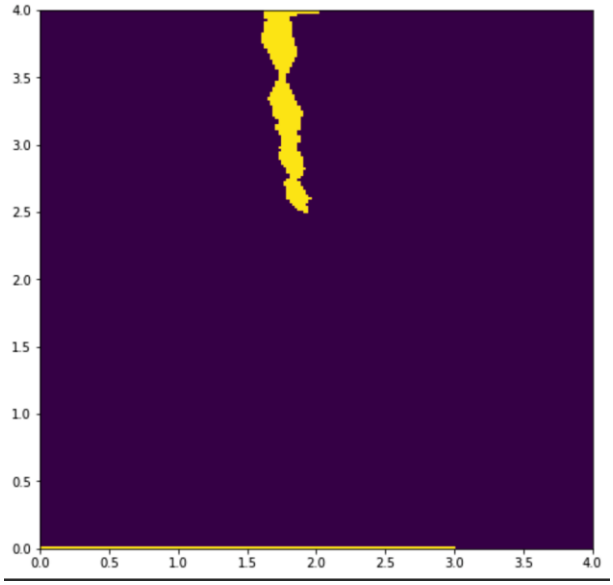

Рисунок 4.4 - Дійсна область Парето

Після використання моделі для сітки на множині $[0,1]^2$ та відовлення до оригінальної множини $[0,4]^2$. Результати наведено нижче (рис. 4.5)

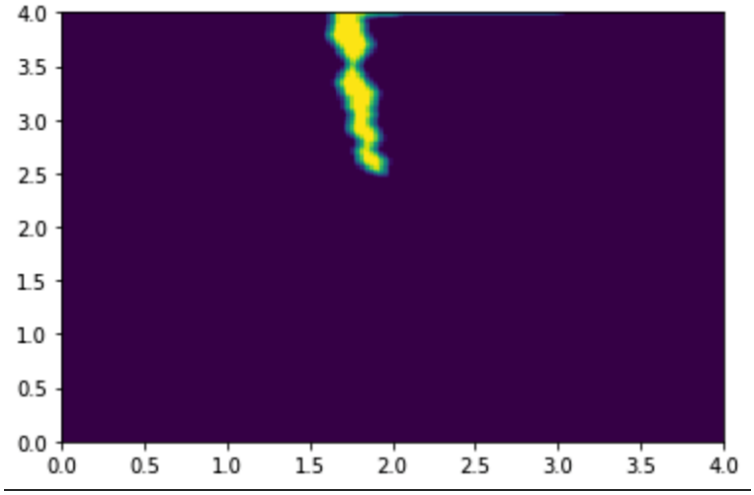

Рисунок 4.5 - Зформована множина Парето

В порівнянні із справжньою множиною Парто можна відмити дві наявні проблеми під час відновлення. По-перше, втрачено область, що відповідала

низьким значенням одного з аргументів. Такий ефект є достатньо природним, якщо згадати, що наше наближення будувалося на базі площі. По-друге, межа області, що розміщена є близькою до кусково-лінійної функції, що може бути покращено для суттєво нелінійних функцій.

Таким чином, застосуємо фільтрацію, що базується на наступній ідеї: Якщо для кожної точки, що розташована на межі множини Парето (перевіряється кількість сусідів із множини Парето) підрахувати відносну чутливість щодо зміни на фіксоване епсілон, то можна віднайти розмір кола в просторі аргументів. Таким чином видаляємо випадково точки у напрямку найменшого поширення області Парето. Ця процедура зменшує результуючу область, проте зберігає область Парето. Після застосування фільтрації на основі константи Ліпшиця маємо (рис. 4.6):

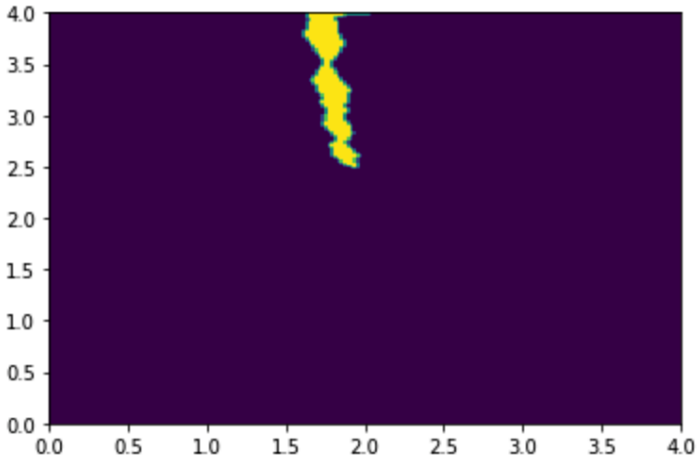

Рисунок 4.6 - Фільтрована множина Парето

Таким чином дійсно фільтрація збільшила якість зформованої множини Парето.

4.4 Порівняння для методів фільтрації

В другому розділі було описано декілька методів встановлення відповідності між областями значень та визначень системи. Ці методи, як вже було показано в попередньому підрозділі, фільтрація на основі віднайдення чутливості функціональних залежностей, дозволяє покращити формування множини Парето. В даному підрозділі буде проведено порівняння запропонованих у другому розділі методів для формування множини Парето.

Накращим методом фільтрації виявився метод на основі підходу із використанням методу мішаного лінійного програмування, інші методи надали гірші результати.

4.5 Висновки четвертого розділу

Отже в четвертому розділі було продемонстровано процес навчання кожної компоненти описаної у попередньому розділі, що виявився задовільним для кожної компоненти. Як результат, отриману модель було протестовано не невідомих функціональних залежностях, що показало також непогані результати. Нарешті, було описано процес фільтрації та якого застосування для застосування після моделі сегментації.

Було проведено порівняльний аналіз кожного із методів фільтрації. Виявилося, що найкращим є метод, заснований на застосуванні мішаного лінійного програмування.

РОЗДІЛ 5: СТАРТАП ПРОЕКТ

5.1 Вступ та постановка задачі стартап проекту

В контексті сучасних реалій вкрай важливим є створення складних технічних виробів із великою кількістю внутрішніх та зовнішніх показників. Як вже зазначалося, внутрішні показники можуть бути попередньо налаштовані із метою оптимального проектування зовнішніх показників виробу. Незважаючи на те, що технології грають все більшу роль в проектуванні таких виробів, велика кількість підприємств все ще схильна або ж до мануального налаштування або ж до вибору далеко не найбільш оптимального плану.

На ринку існує велика кількість конкурентів, що пропонує професійний аналіз та налаштування, проте більшість з цих компаній пропонує або ж ексклюзивні в ніші послуги, або використовує складні в реалізації методи та є довгостроковими. На практиці, в багатьох ситуаціях, важливим є своєчасне проведення аналізу.

Сучасні методи машинного навчання дозволяють пошук множини Парето за достатньо малих часових затрат та використання отриманих результатів для забезпечення найкращого ефективного проектування складних технічних виробів. Отже до задач даного розділу варто передусім віднести маркетинговий аналіз:

1. Створення бізнес-ідеї проекту, виокремлення основних напрямків розвитку технології, описати можливі сильні, слабкі та нейтральні сторони в проекції на можливих конкурентів.

2. Проведення аналізу ринкових можливостей та розробка стратегій виведення та впровадження технології на ринкове середовище.

Створення власне стартап-проекту, що включає наступні кроки:

1. Визначення обсягу необхідних витрат.

2. Розрахунок основних фінансово-економічних показників проекту (такі як собівартість, податковий збір, чистий прибуток та ціна продукту). Визначення показників інвестиційної привабливості продукту.

3. Встановлення першочергових ризиків, що постають загрозою для становлення проекту, аналізування можливих шляхів запобігання ним.

Нарешті, остаточно, даний розділ ставить за мету описати та навести потенційні можливості комерціалізації проекту, а саме:

1. Визначення цільової групи інвесторів, їх бізнес інтересів.
2. Створення та компонування інвестиційної пропозиції.
3. Визначення та просування оферти інвесторам.

Нижче опишемо реалізацію окреслених вище кроків.

5.2 Карта стартап проекту

Стартап представляє з себе побудову та інтеграцію системи, ціль якої є автоматизація узгодження та показників виробів, за заданої множини обмежень на показники.. Нижче наведено карту стартапу (таблиця 5.1):

Таблиця 5.1 - Інформаційна карта проекту

1. Назва проекту	Система побудови допустимої множ параметрів гарантованого узгодже функціонування приладу.
2. Автори проекту	Шарапов Євгеній
3. анотація	Система дозволяє віднайти множ Парето заданих внутрішніх параметрів

Продовження таблиці 5.1

4. Термін реалі проекту	3 місяці
5. Необхідні ресурси	Обладнання - комп’ютер із під’єднан до мережі Інтернет, хмарний сервіс навчання моделей, зберігання да Фінансові ресурси - заробітна п працівникам на 3 місяці, витрати оплату комунальних послуг. 2-кімн приміщення.
6. Опис проблеми, вирішує проект	Дана система дозволяє віднайти множину вхідних параметрів проектування складного технічн вибору, що відхилення від задан налаштування не дозволить одночасн покращення всіх вихідних показникі
7. Головні цілі завдання проекту	оловна мета проекту - створе хмарного сервісу, що віднаходит множину Парето.
8. Очікувані результа	Прикладне програмне забезпечення, розміщується в хмарному середов AWS, та може бути використано онл для отримання зазначеної інформації

5.3 Технологічний аудит

Таблиця 5.2 - Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
Використання методів встановлення відповідності між вхідними та вихідними параметрами в якості попереднього застосування в онлайн сервісах з проектування складних виробів для побудови множини Парето.	Проектування виробів	Дозволяє завчасно звузити області вибору параметрів налаштування виробу
	Пост-аналіз виробів метою пошуку помилок реалізації	Дозволяє встановити наявність помилок, що були допущені при попередньому проектуванні.
	Прогнозування вихідних характеристик при заданих вхідних характеристиках	Модель, що використовується під час побудови області Парето може бути для прогнозування
	Оптимальне використання наданих ресурсів	Можливість проведення найекономічнішого проектування виробу із

Нижче наведемо аналіз потенційних техніко-економічних переваг ідеї в порівнянні із існуючими пропозиціями конкурентів. Цей процес включає в себе попередній перелік техніко-економічних властивостей та характеристик ідеї, виокремлення конкурентів та власне проведення порівняльного аналізу. В Україні конкуренти є достатньо активними, оскільки на сьогоднішній день напрями розвитку, що відповідають обраному є вельми перспективними, тому будемо розглядати 3 конкуруючи продукти схожого напрямку, реалізовані в межах українських розробників.

В першій таблиці буде проведено порівняльний аналіз сильних, слабких та нейтральних сторон продуктів-конкурентів. В наступній таблиці буде проаналізовано технічну здійсненність заданої технології.

Наведемо результати нижче (таблиця 5.3):

Таблиця 5.3 - Визначення сильних слабких та нейтральних характеристик проекту.

№	Техніко-економічні показники ідеї	*потенційні товари/концепції конкурентів*				*W (слабка сторона)*	*N (нейтральна сторона)*	*S (сильна сторона)*
		Мій проект	APRO SOFTWARE	Zfot Group	It-Jim			
1	Точність відновлення	Висока	Висока	Висока	Власний алгоритм			+
2	Тривалість виконання	Середня	Висока	Висока	Середня		+	
3	Безпека даних	Безпечн	Безпечн	Безпечн	Безпечн			+

		ий	ий	ий	ий			

Продовження таблиці 5.3

4	Навантаження на сервер	Високе	Середнє	Середнє	Середнє	+		
5	Доступність для кінцевого користувача	Середня	Висока	Висока	Середня		+	
6	Складність використання	Низька	Середня	Середня	Низька			+

Нижче наведемо аналіз доступності та можливості технічно здійснити виконання технологічної концепції (таблиця 5.4):

Таблиця 5.4 - Технологічна здійсненність проекту

№	Ідея проекту	Технологія реалізації	Наявність технології	Доступність технологій
1	Створення хмарного середовища для аналізування узгодженості внутрішніх показників системи для формування області Парето множини критеріїв	Мова програмування C++	Наявна	Не доступна
2		Мова програмування Python	Наявна	Доступна
3		Мова програмування R	Наявна	Доступна

5.4 Аналіз ринкових можливостей запуску стартап-проекту

Перш за все важливим є проведення попереднього аналізу ринку даних послуг із метою встановлення потенційної складності входження у дани нишу і можливості зайняти на ринку відповідних послуг. Результати даного аналізу наведено нижче: (таблиця 5.5)

Таблиця 5.5 - Попередня характеристика потенційного ринку стартап проекту.

	Показники стану ринку (найменування)	Характеристика
	Кількість головних гравців на ринку (од.)	4
	Загальний обсяг продаж	2 млн. $
	Динаміка ринку (якісна оцінка)	Зростає
	Наявність обмежень для входу	Немає
	Специфічні вимоги до стандартизації та сертифікації	Немає
	Середня норма рентабельності в галузі (або по ринку), %	7%

Зважаючи на кількість гравців на ринку та інші показники ринку, можна зробити висновки щодо сприятливості входження із метою реалізації запропонованого в роботі програмного продукту. Так, динаміка є зростаючою, а конкурентів достатньо небагато.

За наступне завдання поставимо собі характеризацію основних груп потенційних користувачів продукту: (таблиця 5.6)

Таблиця 5.6 - Характеристика потенційних клієнтів стартап-проекту

№	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
1	Покращення технологічних характеристик виробів	Закоренілі на ринку підприємства	Великий досвід у веденні власного бізнесу, значний початковий капітал	Простота та прозорість у використанні
2	Заощадження економічних ресурсів	Нові гравці на ринку виробництва певних виробів	Відкриті до інноваційних підходів молоді спеціалісти	Ефективність на тестових прикладах.
3	Довгострокове покращення підприємства	Великі інвестори	безпека та відсутність загрози для власного бізнесу	Надійсність в обробці даних, гарантованість результату

Нижче проаналізуємо можливі загрози, що можуть виникнути в умовах ринкової конкуренції, занотуїмо їх нижче: (таблиця 5.7)

Таблиця 5.7 - Фактори загроз

№	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1	Збут	Складність адаптації технології до українського ринку	Дослідження нюансів збуту на українському ринку, дослідження систем статуту
2	Конкуренція	Наявність достатньо серйозних конкурентів та їх непередбачуваність	Постійний моніторинг конкуруючих компаній, пошук нових кадрів, що здатні швидко реагувати на зміни на ринку
3	Застаріле обладнання	Домінантна кількість застарілого обладнання підприємств	Розпочати із нових невеликих підприємств, що є більш інноваційно-охочими
4	Інтеграція	Проблема ефективної інтеграції у виробничі процеси підприємства	Проведення консультацій спеціалістів окремих підприємств.

Далі опишемо наявність можливості: (таблиця 5.8)

Таблиця 5.8 - Фактори можливостей

№	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*

1	Вихід на міжнародний ринок	Одночасний вихід міжнар. та україінський	Розширення штату працівників, дослідження

Продовження таблиці 5.8

2	Поєднання зусиль із іншими компаніями	Потенційне поєднання з іншими схожими компаніями для формування коаліції	Створення коаліції та проведення узгодження побудованих сервісів
3	Покращення показників сервісу	Проведення покращення існуючого програмного забезпечення	Потенційне підвищення конкурентоспроможності на ринку

Тепер проведемо характеризацію наявного конкурентного середовища, до цього етапу можна передусім віднести визначення типу та рівню конкуренції, результати можна побачити знизу: (таблиця 5.9)

Таблиця 5.9 - Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства*
Тип конкуренції: Досконала конкуренція	Велика кількість компаній, що надають послуги	Розробити продукт, що може мати конкурентоспроможні якості
За рівнем конкурентної боротьби: В межах однієї країни	Ринок функціонує за використання вітчизняних потужностей	Обмежитися, що охоплює виключно українських підприємців
За галузевою ознакою: внутрішньогалузева	-	-

Конкуренція за видами товарів: товарно-родова	Конкуренція між моделями	Підвищення показників моделювання

Продовження таблиці 5.9

За характером конкурентних переваг: Нецінова	Показники моделювання залежать суто від налаштування моделі	Покращення моделей
За інтенсивністю: марочна	Бренд має велике значення	Розвиток бренду

Звернемо тепер увагу до моделі 5 сил конкуренції Майкла Портера, вона дозволяє всебічно охопити структуру галузі, проаналізувати її та оцінити привабливість з точки зору отримання прибутку, оцінки конкуренції та ,нарешті, окреслення бізнес-плану. Нижче наведемо результати (таблиця 5.10):

Таблиця 5.10 - Аналіз конкуренції в галузі за М.Портером

Складові аналізу	*Прямі конкуренти в галузі*	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*
	Аналогічні системи	Цінова політика, якість моделювання, привабливість реклами, закоріненість на ринку	Доступність в ціновому сенсі	Опитування, система фідбеків, коментарі та оцінки на аналогічні послуги на ринку	Цінова, технологічна, привабливість споживачам
Висновки	Інтенсивна конкуренція може розпочатися в наближчому майбутньому	Ціна на товар є більш привабливою для споживачів, необхідно опрацювати рекламну компанію	Пряма необхідність в постачальниках відсутня	Клієнти зацікавлені в більш гнучких та економічно привабливих товарах	Наявна невелика кількість, що пропонують більш якісні проте цінова складова є менш

					привабливою

Таким чином конкурентна середа є привабливою на ринку, оскільки існуюча кількість конкурентів пропонує хоча і покращені товари, проте пропонує їх за більш високу ціну. Існування передумов для запуску стартап проекту підтверджується таблицею 5.10, беручи до уваги характеристики ідеї стартап проекту (таблиця 5.5) та вимоги до до продукту (таблиця 5.6). Властивості ринкового середовища були сформовані (таблиці 5.7 та 5.8), що дозволяє оформити наступний аналіз (таблиця 5.11):

Таблиця 5.11 – Обґрунтування факторів конкурентоспроможності

Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
Низька конкуренція	Виявлено невелику кількість конкурентів
Доступність	Незалежність від платформи, побудован на основі хмарних обчислень
Зручність у використанні	Нескладно використовувати

Виділяємо сильні та слабкі сторони на основі таблиці 5.12

Таблиця 5.12 - Таблиця порівняння з конкурентами

№	*Фактор конкурентоспроможності*	*Бали*	*Рейтинг товарів-конкурентів*

1	Точність моделі	19	-1
2	Час обробки	17	-3
3	Навантаження	21	+1

Спираючись на описані дані, можемо провести SWOT аналіз, результати якого дозволяють проаналізувати сильні, слабкі сторони, можливості та загрози для бізнесу: (таблиця 5.13):

Таблиця 5.13 – SWOT- аналіз стартап-проекту

Сильні сторони: Невелика кількість конкуренті Достатньо низьке навантаженн непогана точність,	Слабкі сторони: Сильні конкуренти, Недостатньо високі стандарти безпеки.
Можливості: Вихід на міждународний ринок Інтеграція із іншими сервісами	Загрози: Втрата приватних даних клієнт Ззіпсування репутації за рахун того, що модель не ідеальна

На основі проведеного аналізу, що його було занотовано у таблицю 5.13, можемо зробити висновок та спроектувати альтернативи ринкового впровадження стартап проекту, занотуємо їх у таблицю 5.14:

Таблиця 5.14 – Альтернативи ринкового впровадження стартап проекту

№	*Альтернатива ринкової поведінки*	*Ймовірність отримання ресурсів*	*Сроки реалізації*
1	Стрімкий вихід на ринок із заниженою швидкістю та якістю формування	20%	1 місяць
2	Поступовий	45%	3 місяці

	вихід		

Отже, базуючись на проведеному аналізі та наведених міркуваннях, заноованих у таблиці, робимо висновок про сприятливість ринкових умов для впровадження технології та доцільності виходу на ринок.

5.5 Розроблення ринкової стратегії проекту

Перш за все поставимо собі за мету визначити та окреслити стратегію охоплення ринку. Для цього ключовим є аналізування груп потенційних споживачів. (таблиця 5.15):

Таблиця 5.15 - Вибір цільових груп потенційних с

№	Опис профілю цільової гру потенційни клієнтів	Готов ь споживачі прийняти продукт	Орієн ний попит в межах цільо групи	Інтенс ість конкуренції сегменті	Прост входу у сегм
1	Закор і на ринку підприємст	Низьк	7%	Низьк	Серед
2	Нові гравці на ри виробництв певних вир	Висок	20%	Висок	Низьк
3	Велик інвестори	Низьк	10%	Висок	Серед
Обрані цільові групи: 2, 3					

За стратегію охоплення ринку було обрано масовий маркетинг — компанія концентрує свої зусилля одразу на всіх сегментах споживачів. Для цього сформуємо базову стратегію розвитку (таблиці 5.16 - 5.18):

Таблиця 5.16 - Визначення базової стратегії розвитку

№	*Обрана альтернатива розвитку*	*Стратегія охоплення ринку*	*Ключові конкурентноспроможні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку*
1	Перша та друга	Стратегія недиференційованого маркетингу	Велика кількість галузей, універсальність	Масовий маркетинг

Таблиця 5.17 - Визначення базової стратегії конкурентної поведінки

Чи є проект першопрохідцем на ринку?	Чи шукатиме компанія нових клієнтів чи забиратиме у конкурентів	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки*
Ні	Так	Так	Стратегія виклику лідера

Таблиця 5.18 - Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	*Базова стратегія розвитку*	*Ключові конкурентоспром ожн і позиції власного стартап-проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)*
Якість моделей, безпека даних	Відносна легкість у використанні, невибагливість та приваблива ціна	Якість прогнозу, універсальність	Універсальність та необхідність у виробництві

5.6 Висновки до четвертого розділу

Таким чином під час проведення дослідження було створенно товар, що представляє техніко-економічну цінність для споживачів на ринку, є конкурентоспроможним. Проведений у розділі аналіз дозволяє сміливо стверджувати про готовність стартапу до виходу на ринок, його економічну зрілість.

ВИСНОВКИ

Отже в магістерській дисертації було детально розглянуто проблему формування множини Парето на основі попереднього формування невідомих функціональних залежностей та узгодження областей значень та визначення. Було запропоновано низку підходів для досягнення поставленої задачі узгодження, а також запропоновано підхід до навчання та використання архітектури для формування множини Парето.

Формування функціональних залежностей в ,передусім, адитивному чи мультиплікативному вигляді є першим кроком до проведення відповідного аналізу. Таким чином чисельні методи, методи аналізу даних та алгоритми машинного навчання приходять на допомогу. Використання цих підходів було детально описано у першому розділі.

Оскільки поставлена задача є достатньо складною проблемою, що вимагає формалізації, її розв’язок проводитемо поетапно, де на першому етапі було поставлено за задачу розроблення методів відновлення залежності між областю значень та визначень відповідних відомих функціональних залежностей. Було запропоновано низку методів, серед яких метод зведення задачі до задачі лінійного змішаного програмування, наближення константи Ліпшиця та побудови та навчання особливої нейронної мережі.

Нарешті, у третьому та четвертому розділі було детально розглянуто архітектуру на основі використання сегментаційних моделей та автокодувальників із подальшим використанням підходів другого розділу.

Остаточним кроком було описання можливого стартапу, що може бути створений на основі запропонованої технології. В результаті проведеного дослідження, виявилося можливим створення стартапу на основі запропонованої моделі.

ПЕРЕЛІК ПОСИЛАНЬ

1. В. В. Янковий, Д. Л. Стефанович Київська політехніка: початок історії. Київ: КПІ ім. Ігоря Сікорського, Вид- во «Політехніка», 2018. 244 с.

2. Панкратова Н.Д. Системная оптимизация конструктивных элементов современной техники. *Киберн. и системный анализ*. 2001. №3. С.119-131.

3. Pankratova, N., Maistrenko, O., Maslianko, P. System definition of the business/enterprise model. In: *P. Herrero, H. Panetto, R. Meersman, T. Dillon (eds.) On the Move to Meaningful Internet Systems: OTM 2012 Workshops, Lecture Notes in Computer Science*, vol. 7567, pp. 134-143. Springer Berlin / Heidelberg, 2012.

4. Pankratova, N., Maistrenko, O., Maslianko, P.: Business model for analysis of the university research and scientific collaboration: A case study// http://link.springer.com/chapter/10.1007/978-3-642-38366-3_5 201.

5. Панкратова Н.Д., Опарина Е.Л. Формирование множества Парето в задачах поиска рационального компромисса. Сб. Научных трудов XXII Международной научно-практической конференции *«Системный анализ в проектировании и управлении»*. (SAEC-2020).– Санкт-Петербург, Политех-пресс. - 13-14 октября, 2020. – С.66-77. УДК 007.52:519.6:539.3:681.3 doi:10.18720/SPBPU/2/id20-111

6. Nathan Marz. Big Data: Principles and best practices of scalable realtime data systems, Mayne: Manning publicastions, 2015, 215 p.

7. Панкратова Н.Д. Системний аналіз. Теорія та застосування. Київ: Наукова Думка, 2018. 348 с.

8. Панкратова Н.Д. Формирование множества Парето в системной задаче концептуальной неопределенности. *Доповіді НАН України*. 2001. № 12. С. 65 - 70.

9. Cybenko, G. Approximation by Superpositions of a Sigmoidal function. *Mathematics of Control Signals and Systems*. 1989. №2. С.303 - 314.

10. Murphy Kevin P. Machine Learning: A Probabilistic Perspective. Michigan: MIT Press, 2012. 1096 p.

11. R. S. Sutton and A. G. Barto. Reinforcement Learning: An Introduction. The MIT Press, 2 nd edition, 2018. 796 p.

12. Michele Conforti, Gérard Cornuéjols, Giacomo Zambelli Integer Programming. New York: Springer Cham Heidelberg. 2014. 466 p.

13. Cem Anil, James Lucas, and Roger Grosse. Sorting out Lipschitz function approximation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*. volume 97 of Proceedings of Machine Learning Research, pages 291–301, Long Beach, California, USA, 09–15 Jun 2019.

14. Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. *In Advances in Neural Information Processing Systems*, 2017. P 6240–6249

15. Yurii Nesterov. Introductory lectures on convex optimization: A basic course, volume 87, New York: Springer Science & Business Media, 2013.

16. Tsui-Wei Weng et. al. Towards fast computation of certified robustness for relu networks. arXiv preprint arXiv:1804.09699, 2018.

17. Huan Zhang et. al.. Efficient neural network robustness certification with general activation functions. *In Advances in Neural Information Processing Systems*, volume 45, 2018. P. 4939–4948

ДОДАТОК А: ПРОГРАМНИЙ КОД

```
import numpy as np
import pandas as pd
from tqdm import tqdm

import torch
try:
  import torchmetrics
except:
  %pip install torchmetrics
  import torchmetrics

from torch import autograd
from torch import nn
from torch.nn import functional as F
from torch.utils.data import TensorDataset, Subset, DataLoader, random_split

from sklearn.model_selection import train_test_split

from matplotlib import pyplot as plt

from sklearn.svm import SVC, LinearSVC
from sklearn.metrics import accuracy_score

from functools import reduce
```

```
from itertools import product, chain, accumulate
try:
    import pulp
    import cvxopt
except:
    %pip install pulp
    %pip install cvxopt
finally:
    from pulp import LpInteger, LpMaximize, LpMinimize, LpProblem, LpVariable, value
    from cvxopt import solvers, matrix
A = np.array([])

for i, child in enumerate(model.children()):
    if isinstance(child, nn.Linear):
        print(i)
        u, l = level_bounds[i][1].detach().numpy(), level_bounds[i][0].detach().numpy()
        block = np.block([
         [-np.eye(u.shape[0]), np.eye(u.shape[0])],
         [np.zeros((u.shape[0], u.shape[0])), np.eye(u.shape[0])],
         [-u * np.eye(u.shape[0]), (u - l) * np.eye(u.shape[0])]
        ])

        m, n = block.shape
```

```
            indicator = np.array([list(chain.from_iterable([[1, 1, 1] if u[i] > 0 or l[i] <
0 else [0, 0, 0] for i in range(len(u))]))]).T
            block = block * indicator

            A = np.block([[A, np.zeros((A.shape[0], n))],
                          [np.zeros((m, A.shape[1])), block]]) if len(A) else block


        print(A.shape)
        def get_level_bounds(model, lower_bounds, uppper_bounds, n=1000):

            """
            Estimate the upper bounds of the intermediate layers by
            uniformly sampling from the domain of the input variables and
            pushing them through the network layerwise
            """

            dist = torch.distributions.Uniform(torch.tensor(lower_bounds).float(),
torch.tensor(uppper_bounds).float())
            samples = dist.sample((n,))
            levels = []

            x = samples

            for child in model.children():
```

```
            x = child(x)
            levels.append((x.min(0)[0], x.max(0)[0]))

        return levels

    def define_safe_delta(model, point, lower_bounds, upper_bounds):

        """
        Accomplishes a mixed integer programming to derive the addmissible interval
        width for the specified point of the fucntion domain wrt fitted nn model
        """

        level_bounds = get_level_bounds(model, lower_bounds, upper_bounds)

        children = list(model.children())
        n_x = int(len(children) / 2) + 1
        n_y = int(len(children) / 2)

        n = sum([x.in_features + x.out_features for x in children if isinstance(x, nn.Linear)])

        P = 2 * np.eye(n) * np.eye(n) * np.array([1 if 0 <= i < children[0].in_features else -1 \
```

```
                                    if n - children[-1].out_features < i <= n - 1 else 0 \
                                        for i in range(n)]).reshape(-1, 1)
            q = 0

            #inequalities
            A = np.array([])
            for i, child in enumerate(model.children()):
                if isinstance(child, nn.Linear):

                    u, l = level_bounds[i][1].detach().numpy(), level_bounds[i][0].detach().numpy()

                    block_mat = np.block([
                     [-np.eye(u.shape[0]), np.eye(u.shape[0])],
                     [np.zeros((u.shape[0], u.shape[0])), np.eye(u.shape[0])],
                     [-u * np.eye(u.shape[0]), (u - l) * np.eye(u.shape[0])],
                    ])

                    block_b = np.block([
                        [np.zeros((u.shape[0], u.shape[0]))],
                        u * l * np.eye(u.shape[0]),
                        [np.zeros((u.shape[0], u.shape[0]))],
                    ])

                    #the problem with the equality constraints, inequalites are tackled
```

```
            m, n = block.shape
            indicator = np.array([list(chain.from_iterable([[1, 1, 1] if u[i] > 0 or l[i] < 0 else [0, 0, 0] for i in range(len(u))]))]).T
            block = block * indicator
            b = b * indicator

            A = np.block([[A, np.zeros((A.shape[0], n))],
                          [np.zeros((m, A.shape[1])), block]]) if len(A) else block


        return A
    hdims = (128, 256)
    batch_size, epochs, lr = 1, 500, 1e-5
    epochs = 1000

    dataset = TensorDataset(X, y)
    dataset_train, dataset_val = random_split(dataset, lengths=[int(0.8 * n), n - int(0.8 * n)])
    dataloader_train, dataloader_val = DataLoader(dataset_train), DataLoader(dataset_val)

    model = nn.Sequential(

        nn.Linear(1, hdims[0]),
```

```
            nn.ReLU(),
            nn.Linear(hdims[0], hdims[1]),
            nn.ReLU(),
            nn.Linear(hdims[1], 1),

        )

        criterion_regression = nn.MSELoss()
        optimizer_regression = torch.optim.Adam(lr=1e-3,
params=model.parameters())
        hparams_svm = {
            'samples': 100,
            'step': 1e-3,
            'lr': 1e-3,
            'eps': 1e-3,
        }

        hparams={
            'epochs': epochs,
            'eps': 1e-1,
            'input_dim': k,
            'samples': 100,
            }

        dataloaders = {
```

```
    'train': dataloader_train,
    'val': dataloader_val,
  }


  #collate_fun =
  def sample(samples, radius, center, sampler):

    """ samples uniformly from the euclidian ball with the center at @param center
      with the radius of @param radius
    """

    samples_radius = radius * sampler.sample((samples,))
    samples_
    radius = torch.distributions.uniform.sample(0, radius, )

  def train_model(model, dataloaders, optimizer, criterion, hparams):

    running_train_mse = []
    running_val_mse = []

    epochs = hparams['epochs']
    dataloader_train, dataloader_val = dataloaders['train'], dataloaders['val']
```

```
for epoch in range(epochs):

#Train
  model.train()
  optimizer.zero_grad()
  loss_reg = model_step(dataloader_train, model, optimizer, criterion)
  loss_reg.backward()
  running_train_mse.append(loss_reg.detach().cpu().item())
  optimizer.step()

  #Validation
  with torch.no_grad():

    model.eval()
    loss_reg = model_step(dataloader_val, model, optimizer, criterion)
    running_val_mse.append(loss_reg.detach().cpu().item())

  print(
        f"""
        epoch: {epoch}
        Training mse: {running_train_mse[-1]}
        Validation mse: {running_val_mse[-1]}
        """
    )
```

```
    return model


def train_autoencoder(encoder, decoder, model, dataloaders, optimizer, criterion, hparams):

    dataloader_train, dataloader_val = dataloaders['train'], dataloaders['val']
    metric = torchmetrics.Accuracy()
    epochs, eps, samples = hparams['epochs'], hparams['eps'], hparams['samples']
    dataloader_train, dataloader_val = dataloaders['train'], dataloaders['val']

    #sampling uniformly from the unit euclidean ball
    sampler_vector = torch.distributions.MultivariateNormal(torch.zeros(1), torch.eye(1))
    sampler_normalizer = torch.distributions.Exponential(rate=1)

    running_train_loss = []
    running_val_loss = []

    running_train_acc = []
    running_val_acc = []

    deltas_train = []
    deltas_val = []
```

```
y_hat_mean_train = []
y_hat_mean_val = []

metric = torchmetrics.Accuracy()

for epoch in range(epochs):

  running_train_loss.append([])
  running_val_loss.append([])

  running_train_acc.append([])
  running_val_acc.append([])

  deltas_train.append([])
  deltas_val.append([])

  y_hat_mean_train.append([])
  y_hat_mean_val.append([])

  encoder.train()
  decoder.train()

  #First train the network to approximate the outputs
```

```
        #tqdm_batch = tqdm(enumerate(dataloader_train), desc='training', position=0)

        for ix, batch in enumerate(dataloader_train):

            optimizer.zero_grad()

            x, y = batch
            x, y = x.float(), y

            #delta = encoder(x)
            #print("kek")
            delta = eps / (encoder(x))
            z_, Y = sampler_vector.sample((samples,)), sampler_normalizer.sample((samples,))

            z = z_ / ((Y  + (z_ ** 2).squeeze()) ** 0.5).unsqueeze(1)
            x_perturbed = delta.squeeze() * z + x

            #print(z.shape, x.shape)

            y_hat = model(x_perturbed)

            target = binarize_target(y_hat, y, eps)
```

```
            #print(target.shape, y_hat.shape)
            loss_ae = 0.5 * F.kl_div(y_hat, torch.from_numpy(np.random.uniform(-
eps, eps, y_hat.shape)).float()) - 0.5 * delta
            #loss_ae = 0.3 * criterion(y_hat, target.squeeze()) + 0.3 *
y_hat[:,0].mean() + 0.4 * delta
            acc_score = close_metric(y_hat, y, eps)

            loss_ae.backward()

            optimizer.step()

            running_train_loss[-1].append(loss_ae.detach().cpu().item())
            running_train_acc[-1].append(acc_score)
            deltas_train[-1].append(delta.detach().cpu())
            y_hat_mean_train.append(y_hat[:,0].mean().detach().cpu().item())

            #tqdm.update(1)
            #if (ix + 1) % 10 == 0:
            print(
                f"""
                epoch: {epoch},
                batch: {ix},
                Training loss: {np.mean(running_train_loss[-1])},
                Training accuracy: {np.mean(running_train_acc[-1])},
                Mean deltas: {np.mean(deltas_train[-1])},
```

```
            Mean logits: {np.mean(y_hat_mean_train[-1])},
            """
        )


    def model_step(dataloader, model, optimizer, criterion):

        """
        A general step for training decoder model (before using one in the encoder
        training
        """

        X, Y = dataloader_train.dataset[:]
        X, Y = X.float(), Y.unsqueeze(1)
        Y_hat = model(X)

        loss_reg = criterion_regression(Y_hat, Y)

        return loss_reg

    def train(encoder, decoder, model, dataloaders, optimizers, criterions, hparams,
pretrained=True):

        model = train_model(model, dataloaders, optimizer_regression,
criterion_regression, hparams) if not pretrained else model
```

```
model.eval()
encoder, decoder = train_autoencoder()
```