НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

В.о.завідувача кафедри

_____ Оксана ТИМОЩУК

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

на тему: «Узгодження множини значень складної проектованої системи з областю визначення внутрішніх параметрів»

Виконав:
студент IV курсу, групи КА-64
Чикивдя Олександр Іванович _____

Керівник:
д.т.н., професор
Панкратова Наталія Дмитрієвна _____

Консультант з нормоконтролю:
доцент, к.т.н.
Коваленко Анатолій Єпіфанович _____

Рецензент:
д.т.н., професор
Гуляєв Валерій Іванович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент: Чикивдя Олександр Іванович

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
В.о.завідувача кафедри
_____ Оксана ТИМОЩУК
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Чикивді Олександра Івановича

1. Тема роботи «Узгодження множини значень складної проектованої системи з областю визначення внутрішніх параметрів», керівник роботи Панкратова Наталія Дмитрієвна д.т.н., професор, затверджені наказом по університету від «_25_»_травня_ 20_20_ р. №_1143-с__

2. Термін подання студентом роботи 08 травня 2020 року_____

3. Вихідні дані до роботи

   1. Операційна система Ubuntu 18.04
   2. Частота процесора 2.5 ГГц
   3. Мова програмування C++
   4. Середовище розробки - Qt Creator
   5. Бібліотеки, що використовувалися: Qt, Eigen, QCustomPlot.

4. Зміст роботи

   1. Проаналізувати природу множини Парето
   2. Розробити математичну базу для алгоритму

3. Розробити програмний продукт пошуку множини Парето

4. Виконати економічний аналіз програмного продукту

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1. Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	к.е.н.,доц. Шевчук О.А.	21.04.20	30.05.20

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Пошук необхідної літератури	13.04.20	
2	Аналіз існуючих алгоритмів та методів	17.04.20	
3	Аналіз даних та створення методу обробки	25.04.20	
4	Розробка теоретичної частини алгоритму	28.04.20	
5	Розробка продукту по реалізації алгоритму	3.05.20	
6	Тестування отриманого продукту	11.05.20	
7	Оформлення дипломної роботи	17.05.20	

Студент Олександр Іванович ЧИКИВДЯ

Керівник Наталія Дмитрієвна ПАНКРАТОВА

[1]∗ Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломної роботи.

РЕФЕРАТ

Дипломна робота: 125 с., 40 рис.,6 табл.,2 дод., 8 джерел.

УЗГОДЖЕННЯ МНОЖИНИ ЗНАЧЕНЬ СКЛАДНОЇ ПРОЕКТОВАНОЇ СИСТЕМИ З ОБЛАСТЮ ВИЗНАЧЕННЯ ВНУТРІШНІХ ПАРАМЕТРІВ

Існуючі кіберфізичні системи працюють на базі певних алгоритмів або певних специфічних нейроних мереж, які якимось чином за своїми результатами роботи дуже близькі до концепту кіберфізичних систем. Проте дуже невелика кількість дійсно існуючих систем можна назвати складними системами за методологією, за якої їх намагаються реалізувати.

Справжні системи є надзвичайно складними, такі системи зазвичай є реагентами на деякі події в реальному світі, за чим можна стежити спостерігаючі за значення деяких величин (значень стану системи), які залежать від деякої множини внутрішніх параметрів, одні з яких є керованими, а на інші нажаль людина не має впливу.

В будь якому випадку існує потреба в тому, аби зв’язати певним чином ці дві множини і отримати деяку область для кожного значення внутрішніх елементів та значень системи, аби мати певне уявлення про поведінку системи та мати певне уявлення про взаємозв’язок цих величин. Таку область прийнято називати множиною Парето і ця множина є одним із ключових фігурантів системного аналізу складних систем.

Програмний продукт було реалізовано за допомогою мови програмування C++ та фреймворку Qt, який дозволив створити певний зручний інтерфейс для користувача.

Отримані результати: розроблено продукт дозволяючий для обраної бази даних створювати нейронну мережу, яка виконує опції апроксимуючої функції, навчати її та використовувати для пошуку множини Парето.

ABSTRACT

Thesis work: 125 p., 40 fig., 6 tables, 2 app., 8 sources.

COORDINATION OF THE VALUES SET FOR A COMPLEX DESIGNED SYSTEM WITH THE DOMAIN OF INTERNAL PARAMETERS

Existing cyberphysical systems work on the basis of certain algorithms or certain specific neural networks, which are somehow very close to the concept of cyberphysical systems. However, a very small number of actual existing systems can be called complex systems according to the methodology by which they are ought to be implemented.

Current systems are extremely complex, such systems are usually reagents for some real-world events, which can be observed by observing the values of some quantities (system state values), which depend on a set of internal parameters, some of which are controlled and others unfortunately the person has no influence.

Under any circumstances, there is a need to relate these two sets in some way and to obtain some domain for each value of the internal elements and values of the system, to have some idea of the behavior of the system and to have some idea of the relationship of these quantities. This area is called the Pareto set and this set is one of the key participants in the system analysis of complex systems.

The software product was implemented using the C ++ programming language and the Qt framework, which allowed to create a certain user-friendly interface.

Obtained results: developed product allows to create a neural network for the selected database, which performs the options of the approximating function, to train it and use it to search for the Pareto set.

ЗМІСТ

ВСТУП

Третє десятиліття третього тисячоліття нашого існування дуже добре продемонструвало, що як би сильно людство як вид не намагалося досягти досить великої мети стосовно світового розвитку, дуже велика кількість чинників змушує нас оглядатись назад і не дає нам йти вперед. гарним прикладом такого чинника стала приголомшивша весь світ пандемія коронавірусу.

Тонкість людської оболонки, яка через все більш і більш комфортні умови існування кожне покоління ставала все менш придатною до реальних природних загроз, остаточно показала відсутність готовності до протистояння зовнішнім факторам. Людство як вид стає на порозі подальшого шляху розвитку: зосередити всю свою увагу на проблемах людської складної системи, здоров’я та зробити все, що лише можна, аби не допустити знову таку саму ситуацію як і зараз.

Проте ця проблема є лише голкою в стогу сіна всіх тих проблем, із якими ми стикаємося кожного дня, і ще менше значимою в порівнянні із потенційними загрозами: людству необхідно досягти нової ступені еволюції розвитку як виду, а саме увійти у стан існування, коли людство займається вирішенням задач лише інтелектуального характеру в межах шостого технологічного укладу.

Це означає, що необхідно створити умови існування, коли кожний індивідуум матиме змогу не розв’язувати проблеми логістичного характеру, такі як покупка продуктів, транспортування до інших точок міста або цілого світу, це все має виконуватись автоматично.

1 МНОЖИНА ПАРЕТО

Щоб отримати вирішення сформулювати проблему недостатньо в цьому випадку: дана задача занадто різнопланова. Ми зіштовхнулись з однією із найактуальніших задач системного аналізу: створення математичного, програмного та технічного апарату для створення складної технічної системи, що дозволить розв'язати цю складну проблему. У своїй книзі "Растригин Л. А. Адаптация сложных систем. — Рига: Зинатне, 1981. — 375 с." Растригін приводить визначення складної системи як певного нестабільного, абсолютно самостійного об'єкту, що має нетерпимість до людського впливу та її поведінка на один і той самий вплив може бути різним в залежності від навколишнього середовища, що виключає можливість побудови системи як деякого статичного об'єкту. [1]

Однією із важливих задач системного аналізу складних систем є узгодження множини внутрішніх параметрів із множиною значень системи. Значення цієї задачі для системного аналізу проявляється в тому, що така множина по суті є множиною значень внутрішніх параметрів та значень стану системи, що забезпечує їй так зване "гарантоване функціонування".

В системному аналізі існує термін "множина Парето", сутність якої наступна: це така сукупність множин внутрішніх та зовнішніх параметрів системи, для якої для будь якого значення вектору значень внутрішніх параметрів із множини Парето система прийме значення із множини зовнішніх параметрів множини Парето. Та навпаки: для будь якого значення зовнішніх параметрів множини Парето існує вектор внутрішніх параметрів, в якому зовнішні параметри системи приймають таке значення.

Практичного значення цей метод набуває якщо до пошуку такої множини підійти із точки зору звуження початкових відрізків до задовольняючих тим чи іншим критеріям. Розглянемо таку задачу: на основі звітів про роботу власник певної компанії прийняв рішення щодо вектору розвитку і хоче досягти певної мети. Була запрошена група експертів, які на основі вимог клієнта створили наступну систему: нехай по всій компанії було проведено опитування щодо

того, яка якість роботи в кого буде в залежності від заробітної плати, яку їм будуть виплачувати. Щоб система була якнайменш зашумленою, опитування було анонімним для самих працівників, проте самі експерти знали хто які посаду займає. Це дуже важливо для оцінки значень стану системи, за які беруть очікуваний прибуток компанії, задоволення працівників, задоволення клієнтів компанії тощо.

Наприклад, якщо перевагу по заробітній платі надати більшу SMM-менеджерам, то це може скомпенсувати невдоволення клієнтами компанії логістичним відділом, що має напроти низьку зарплату і гірше ніж вимагає ситуація проводив упаковку товару. Тому досить важливо знати не лише чисельне значення певної змінної, а і її сутність.

2 МЕТОДИ ДОСЛІДЖЕННЯ СКЛАДНИХ СИСТЕМ

2.1 Система як предмет дослідження системного аналізу

В рамках класичного системного аналізу таке поняття як “система” є занадто обширним та нечітким, щоб дати йому точне визначення, бо методологія системного аналізу вимагає від методів розв’язання задачі абсолютної всеосяжності під час розгляду певної проблеми. Таку абсолютно строгу та незмінну ідею розгляду задачі вимагає сама етимологія цього слова. Із грецької мови саме слово “система” можна перекласти як “той, що складається із частин; сполучений” [3,6].

Окрім самого походження слова, ідея системного аналізу, як методологія комплексного погляду на проблему, вимагає побудови в кожній конкретній задачі чітку структуру взаємозв’язків між об’єктами та суб’єктами розглядаємого простору дій та учасників.

Відповідно до концепту системного аналізу, розгляд понять, які будуть застосовуватись до розв’язку кожної конкретної задачі, так само має бути всеосяжним. Справа в тому, що розглядаючи будь-яку частину світу ми завжди будемо зіштовхуватись з неоднорідністю інформації та об’єктів. Це проявляється з будь якої сторони, з якої ми розглядаємо ту чи іншу проблему. Наприклад, класична теорія математичного аналізу будується на аксіомах та визначеннях, які за принципом розгляду тої чи іншої математичної задачі створюють ієрархію понять та об’єктів. Наприклад, множина рекурсивно заданих функцій таких як поліном Чебишева.

Як і простір із чіткими формалізованими поняттями, реалії життя як приклад середовища неформалізованих понять точно так само вибудовують певну ієрархію об’єктів, що зустрічаються нам під час життя. ще з дитинства ми привчаємось до того, що абсолютно весь світ ми ділимо “чорне” та “біле”, оскільки нам так краще сприймати всю реальність. Проте в певний момент ми зустрічаємо інші кольори, наприклад, сірий, і стандартної множини понять нам стає недостатньо. Так ми розбиваємо всю реальність на певні класи об’єктів, які мають одні і ті самі властивості.

Проте розгляд таких класів є задачею, наприклад, дискретної математики, яка займається теорією множин. Системний аналіз у свою чергу розглядає множину об’єктів, які пов’язані одні з одним. Якщо до цього речення додати єдине уточнення, можна сформулювати визначення системи в рамках системного аналізу:

Системою будемо називати множину однорідних за структурою об’єктів, що пов’язані між собою правилами, що задаються простором функціонування [3,6].

Вже на цьому етапі проявляється складність понять, які ми розглядаємо: класична ієрархічна структура передбачає максимально чітке визначення базового елементу середовища. Фізика передбачає, що весь світ створений із квантів, яким дано чітке визначення неподільних елементарних частинок всього існуючого у всесвіті. Ієрархія класів у мові програмування С++ передбачає скільки завгодно складну структуру об’єкта, функціонування якого залежить лише від потужності обчислювального блоку, проте завжди має бути чітке визначення батьківського класу, наслідування від якого є базою для всіх дочірніх. Будь-якій матеріальній чи віртуальній моделі необхідне чітке визначення базового елементу, з якого створені всі об’єкти.

Такої простої та доступної структури для класичних методів розв’язку достатньо, розглядати таку елементарну структуру об’єкта як щось більш складне, ніж є насправді, недоцільно хоча б з точки зору ефективності, проте на жаль, надзвичайно невелика кількість проблем нашого світу неефективно розглядати як моделі такого елементарного рівня як сукупність елементарних базових частинок, оскільки, по-перше, немає необхідного природного апарату, що хоча б на 95% описав би всі процеси одночасно з їх взаємозв’язками, а по-друге, обчислювальні апарати сучасного світу не змогли би впоратись із обчисленнями такого рівня навантаження (рис 2.1).

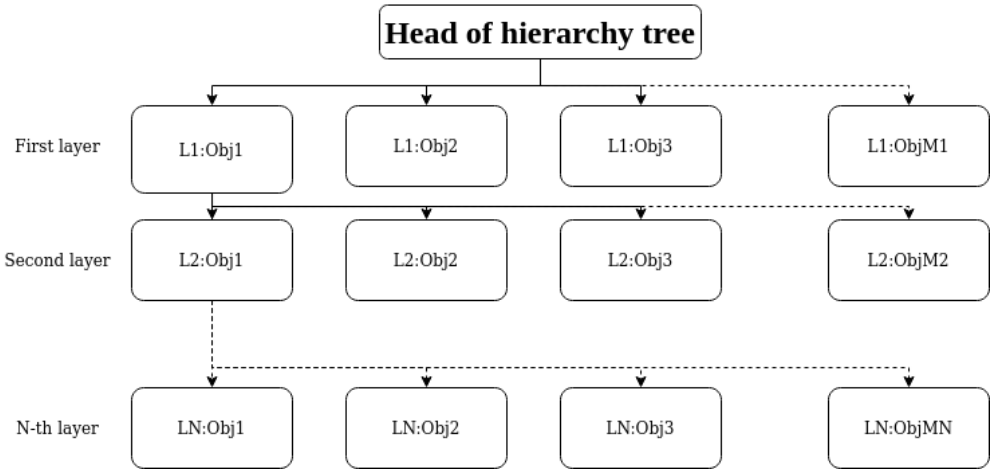


Рисунок 2.1 – Схематичне зображення ієрархічного об’єкту

Наприклад, людське тіло розглядається медициною упродовж багатьох тисячоліть, проте ми майже нічого не знаємо про глибокі структури функціонування людини. Так, ми можемо багато чого сказати про функціональні властивості нашого тіла, виділяти певні закономірності та механізми функціонування, проте загальний процес функціонування людини (не окремі системи як серцево-судинна, нервова тощо, а все тіло загалом) наш мозок поки що не готовий сприйняти навіть наполовину.

Саме тут виникає необхідність розглядати систему як певну сукупність взаємопов’язаних об’єктів невизначеної природи, які неможливо всесторонньо описати зручними та відомими нам методами.

2.2 Складна система як опис структури проблем системного аналізу

Перед тим, як дати визначення складної системі, розглянемо проблематику й причини того, чому описаного визначення системи недостатньо для розв’язання поставлених задач.

Повернемось до прикладу з тілом людини, бо в тому моменті було зачеплено сутність проблеми, чому такого визначення системи недостатньо для апарату системного аналізу. Справжня система, яка зустрічається у 99% випадків, є складнішою, ніж сукупність пов’язаних між собою елементів. Дамо наступне визначення:

Складеною системою називається система, яка складається із різнотипних за структурою об’єктів, які в свою чергу можуть бути системами.

Назва підрозділу підштовхує дати таке визначення саме складній системі, проте є певний ряд властивостей якими має бути наділена складна система для якісного розв'язку задач системного аналізу. Проблеми, яким ця дисципліна має дати способи розв'язання, є занадто різноманітними і, що дійсно важливо, занадто складними, а не складеними.

Якби питання було лише в масивності інформації до опрацювання, то сучасна наука та техніка легко би дала відповідь SQL базами даних, добре оптимізованими бібліотеками Eigen3 мови програмування C++, надзвичайно великим різноманіттям бібліотек мови Python, деякі з яких дозволяють створювати та навчати нейронну мережу в 4 стрічки коду, проте справжні задачі системного аналізу є надзвичайно складними. Мова йде про кіберфізичні системи, складність яких важко усвідомити до кінця навіть вченим цієї галузі системного аналізу. Складна система має бути складеною, проте не будь яка складена система є складною.

Через це для опису складної системи краще використовувати опис що стосується не внутрішньої структури, а зовнішнього прояву та властивостей, які має система.

2.3 Формалізація визначень складної системи

Перед тим як вводити певну формалізацію слід зазначити, що неможливо неперервно розглядати складні системи через їх невідому структуру і ми маємо право розглядати лише апроксимуючі функції до значень вектора стану, тому для розгляду системи необхідно вводити певне розбиття часового відрізку.

Нехай об'єкт функціонує на певному інтервалі $[a,b] \subset R$ та на цьому інтервалі описується вектором стану $Y_t, t \in [a,b]$. Нехай $\{t_i\}_{i=0}^{N_t}$ розбиття інтервалу $[a,b]$ на $N_t + 1$ кроків з кроком $\varepsilon = \frac{b-a}{N_t}$, тобто $\forall i = 0,\dots,N_t : t_i = a + \varepsilon$. Нехай на цьому розбитті задано множини значень вектора стану $\{Y_i\}_{i=0}^{N_t} = \left\{Y_i^0,\dots,Y_i^{N_Y}\right\}_{i=0}^{N_t}$, вектора керуючих параметрів $\{U_i\}_{i=0}^{N_t} = \left\{U_i^0,\dots,U_i^{N_U}\right\}_{i=0}^{N_t}$ та вектора некеруючих параметрів $\{X_i\}_{i=0}^{N_t} = \left\{X_i^0,\dots,X_i^{N_X}\right\}_{i=0}^{N_t}$. Таку трійку множин та часове розбиття називатимемо системою. Будемо використовувати наступне позначення:$S(Y_{N_y}, U_{N_U}, X_{N_X}, [a,b]_{N_t})$.

Нехай задано систему $S(Y_{N_y}, U_{N_U}, X_{N_X}, [a,b]_{N_t})$. Нехай $F_\varepsilon^S : R^{N_X+1} \times R^{N_U+1} \rightarrow R^{N_Y+1}$ - деяка апроксимуюча значення станів системи функція така, що:

$$\forall i \in 0, \ldots, N_t : ||F_\varepsilon^S(X_i, U_i) - Y_i|| < \varepsilon \tag{2.1}$$

Таку функцію будемо називати функцією апроксимуючу систему S з точністю ε, а її k-й елемент функцією апроксимуючою k-й стан системи з точністю ε.

Впродовж роботи із системою слід апроксимувати значення стану функції, оскільки отримані дані, за якими виконується апроксимація, є виключно тестовими. Сама робота складної системи буде динамічною і для оцінювання роботи слід знайти якусь функцію, яка допоможе з певною точністю будувати прогноз під час роботи програми.

Нехай задано систему $S(Y_{N_y}, U_{N_U}, X_{N_X}, [a,b]_{N_t})$. Дана система S називається складеною, якщо:

$$\begin{gathered}\exists \varepsilon > 0 : \forall F_\varepsilon^S : \exists i, j = 0, \ldots, N_Y, i \neq j : \exists k \in 0, \ldots N_t \\ \left| F_\varepsilon^{S(i)}(X_k, U_k) - Y_k^i \right| < \varepsilon, \left| F_\varepsilon^{S(j)}(X_k, U_k) - Y_k^j \right| > \varepsilon\end{gathered} \tag{2.2}$$

Дане визначення слід інтерпретувати наступним чином: існують 2 елементи вектора стану такі, для яких існує деяка точність, на якій до них неможливо застосувати одну і ту саму апроксимуючу функцію, тобто ці дані різнотипні.

Система не повинна мати аналітичних математичних залежностей між внутрішніми параметрами та зовнішніми значеннями попарно чи окремо. Розглянемо як приклад робота в інфекційному відділенні. Як вхідні дані можуть бути картинка, що отримується з камери, показники з датчиків температури, забруднення повітря тощо. Формально це можна записати так:

Нехай задано складену систему $S(Y_{Ny}, U_{N_U}, X_{N_X}, [a,b]_{N_t})$ і $\Gamma = \left\{\Gamma_k | \Gamma_k : R^{N_X+1} \times R^{N_U+1} \rightarrow R^{N_Y+1}\right\}$ - множина всіх аналітичних відображень на множинах заданих розмірностей. Будемо казати, що система є неаналітичною,

якщо виконується наступне:

$$\forall \Gamma_k \in \Gamma, \exists i = 0, \ldots, N_t : \Gamma_k(X_i, U_i) \neq Y_i \tag{2.3}$$

Будь-яка складна система не може бути “чистою” за значеннями. Це відбувається через те, що ми ніколи не можемо дізнатись точних значень тих чи інших її досліджуючих параметрів через велику кількість причин, серед яких можна виділити 2 основні: будь-які значення параметрів системи отримують з датчиків, які дають нам ті чи інші значення з певною похибкою, та як було сказано, задача дослідження складної системи є надто складною для якогось абсолютного опису і існує певна множина процесів, які ми не можемо враховувати під час роботи. Ці всі процеси є другорядними з точки зору теорії керування системою, тому такі процеси зручно вважати шумом. Будь яка така система має цілу низку таких несподіванок, які винуждають дивитися на неї як на складну [1].

Нехай задано складену систему $S\left(Y_{Ny}, U_{N_U}, X_{N_X}, [a,b]_{N_t}\right)$. Складена система $S^*(Y_{N_Y^*}, U_{N_U^*}, X_{N_X^*}, [a,b]_{N_T^*})$ називається продовженням системи S на інтервалі $[a, b+\delta]$, якщо значення множин на цьому інтервалі отримуються за тією ж природою, що і на $[a,b]$, тобто:

$$\begin{gathered}
\forall \varepsilon > 0 : \exists F_\varepsilon^S = F_\varepsilon^{S^*} = F \\
\forall i \in 0, \ldots, N_t : \|F\left(X_i, U_i\right) - Y_i\| < \varepsilon, X_i, U_i, Y_i \in S \\
\forall i \in 0, \ldots, N_t' : \|F\left(X_i, U_i\right) - Y_i\| < \varepsilon, X_i, U_i, Y_i \in S^*
\end{gathered} \tag{2.4}$$

Дане визначення є зворотнім з точки зору визначення складеної системи, тобто це визначення каже, що продовження системи S має ту ж природу, що і S^*.

Нехай складена система $S\left(Y_{Ny}, U_{N_U}, X_{N_X}, [a,b]_{N_t}\right)$ має продовження $S^*(Y_{N_Y^*}, U_{N_U^*}, X_{N_X^*}, [a,b]_{N_T^*})$. Система S буде називатись зашумленою, якщо:

$$'\lim_{\varepsilon \to 0} \sum_{i=N_t+1}^{N_t^*} ||F_\varepsilon^S(X_i, U_i) - Y_i|| \neq 0 \tag{2.5}$$

Це слід інтерпретувати як неможливість ввести абсолютно точну апроксимацію функцій через гіпотетичну зашумленість значень за природою. Через деякий час всі невраховані шуми дадуть про себе знати, наскільки б точно ми не задали апроксимацію на етапі проектування.

Мабуть, що найбільш неприємною для системного аналітика є властивість системи, яку можна описати як неможливість скерувати систему необхідним собі чином, дуже велика кількість чинників впливають на поведінку системи і ці чинники некеровані, а керуючі параметри мають незначний вплив [1].

Нехай складена система $S\left(Y_{Ny}, U_{N_U}, X_{N_X}, [a,b]_{N_t}\right)$ має продовження $S^*(Y_{N_Y^*}, U_{N_U^*}, X_{N_X^*}, [a,b]_{N_T^*})$. Нехай F_ε^S деяка функція апроксимуюча систему S із точністю ε. Дана система буде називатися некерованою, якщо:

$$\forall \varepsilon > 0 : \lim_{N_t^* \to +\infty} P(\forall i \in N_t + 1, \ldots, N_t^* : \exists U_i^* : F_\varepsilon^S(X_i.U_i^*) = Y_i) = 0 \qquad (2.6)$$

Це можна інтерпретувати наступним чином: для вибірки будь-якої довжини, наскільки б точно ми не отримували функцію маючи функцію, що апроксимує значення множин певної системи, збільшуючи кількість спостережень за роботою системи зменшується вірогідність, що будуть існувати керуючі параметри, якими ми можемо отримати необхідні нам результати [1].

Нехай задана складена система $S\left(Y_{Ny}, U_{N_U}, X_{N_X}, [a,b]_{N_t}\right)$. Нехай F_ε^S функція апроксимуюча значення системи S з точністю ε. Дана система називається нестаціонарною, якщо:

$$\exists S^*(Y_{N_Y^*}, U_{N_U^*}, X_{N_X^*}, [a,b]_{N_T^*}) : \exists i \in N_t + 1, \ldots, N_t^* : ||F_\epsilon(X_i, U_i)|| > \varepsilon \qquad (2.7)$$

Дана властивість означає, що певні внутрішні параметри весь час змінюються і система не є стабільною за значенням внутрішніх параметрів, що викликає змінну складність та вимагає перерахунку апроксимуючої функції весь час задля збереження точності. Важливо розуміти, що теоретично вектор є випадковим. Це означає, що провівши один і той самий експеримент кілька разів необов'язково ми

маємо отримати один і той самий вектор результату. Тому можемо сформулювати наступне означення:

Нехай задана складена система $S\left(Y_{Ny}, U_{N_U}, X_{N_X}, [a,b]_{N_t}\right)$. Система називається невідтворною, якщо виконується наступне:

$$\exists S^*(Y_{N_Y^*}, U_{N_U^*}, X_{N_X^*}, [a,b]_{N_T^*}) : \exists i \in N_t+1, \ldots, N_t^*, j \in 0, .., N_t : \\ X_i = X_j, U_i = U_j, Y_i \neq Y_j \tag{2.8}$$

Нехай задана складена система $S\left(Y_{Ny}, U_{N_U}, X_{N_X}, [a,b]_{N_t}\right)$. Якщо дана система є неаналітичною, зашумленою, нестіціонарною, невідтворною та некерованою, то будемо називати її складною.

Такі строгі визначення необхідні хоча бы для того, щоб звузити коло задач, що розглядаються. Дійсно, якщо хоча б одна із властивостей не належить складеній системі, що розглядається, тоді пролема може бути розв'язана і без певних додаткових методів.

Наприклад, якщо певна складена система є аналітичною, тоді існує потужний математичний апарат для розв'язання таких задач, якими б складними вони не були: розв'язки можна отримати за допомогою теорії прийняття рішень або математичного аналізу, чисельні за допомогою теорії оптимізації та досліждення операцій.

Якщо система не є зашумленою, то можна точно ввести апроксимацію даних за теоремою Чебишева про рівномірну збіжність апроксимуючого полінома в вузлах, заданих ним. І задана апроксимація буде точною для майбутніх часових тактів, що значно спростить роботу із системою.

Якщо система є стаціонарною, то для аналізу система ніщо не буде: таким точним як аналіз часових рядів, який спеціалізується на процесах, що мають одну природу і досить непогано працюють із шумами.

Якщо система керована, то проблем із існуванням системи не має бути в будь який момент часу можна знайти вектор параметрів, що буде давати необхідне нам значення.

Найбільш недоречним на перший погляд є необхідність у невідтворності, проте це би означало, що наша система є суто віртуальною і немає ніякого зв'язку із реальним життям, а розгляд суто математичних моделей та систем без якогось відклику у реальному житті не є основною метою досліждння системного аналізу як галузі.

2.4 Узгодження множин та формалізація задачі

Розглянемо три кейси, що дозволять продемонструвати задачу на конкретних прикладах:

Задача 1. Найбільш примітивна та очікувана задача це задача пошуку множини значень критеріїв за зарплатнями певного плану, суть якої це демонстрації найбільш та найменш задовільного результату. Тому створюються початкові границі, в яких власник готовий виплачувати зарплатню і на основі функції, апроксимуючою значення параметрів системи, можна побудувати сітку цих початкових інтервалів та знайти множину значень системи, яку ми можемо отримати користуючись цими сценаріями.

Задача 2. Вже більш цікавою задачею є пошук значень внутрішніх параметрів, в яких досягаються значення критеріїв. Наприклад, власник хоче знайти такий економічний план щодо виплат працівникам, щоб отримати певні результати роботи компанії.

Задача 3. Найбільш цікавим є третя задача, яку можна отримати скомбінувавши перші дві. Ця задача утворюється, коли на виплати працівникам є певний ряд економічних обмежень, а також є конкретні значення критерів роботи, які хочется отримати.

Сформуємо формалізацію такої множини Парето. Нехай задана складна система $S\left(Y_{Ny}, U_{N_U}, X_{N_X}, [a,b]_{N_t}\right)$, що має функцію F_ε^S апроксимуючу систему S з точністю ε, та для кожного вектору системи задані обмеження:

$$\begin{aligned} L_Y &= \{[a_0, b_0]_Y, \ldots, [a_{N_Y}, b_{N_Y}]_Y\} \\ L_U &= \{[a_0, b_0]_U, \ldots, [a_{N_U}, b_{N_U}]_U\} \\ L_X &= \{[a_0, b_0]_X, \ldots, [a_{N_X}, b_{N_X}]_X\} \end{aligned} \tag{2.9}$$

Множиною Парето будемо називати таку трійку $\{P_Y^*, P_U^*, P_X^*\}$:

$$\begin{aligned} P_Y^* &= \{[a_0^*, b_0^*]_Y, \ldots, [a_{N_Y}^*, b_{N_Y}^*]_Y\} \\ P_U^* &= \{[a_0^*, b_0^*]_U, \ldots, [a_{N_U}^*, b_{N_U}^*]_U\} \\ P_X^* &= \{[a_0^*, b_0^*]_X, \ldots, [a_{N_X}^*, b_{N_X}^*]_X\} \end{aligned} \tag{2.10}$$

Якщо:

$$\begin{aligned} &\forall X \in P_X^*, \forall U \in P_U^* : U_\varepsilon(F_\varepsilon^S(X_i, U_i)) \cap P_Y^* \neq 0 \\ &\forall Y \in P_Y^* : \exists X \in P_X^*, U \in P_U : F_\varepsilon^S(X, Y) \subset U_\varepsilon(Y), \end{aligned} \tag{2.11}$$

де U_ε — ε-окіл точки.

2.5 Класичні апроксимуючі функції

Основною проблемою під час розв'язання задач системного аналізу полягаючої в дослідження складної системи є проблема, пов'язана із пошуком апроксимуючої функції. Дійсно, як стверджувалось раніше, все було б значно простіше, якби система не була неаналітичною. Із неаналітичними системами занадто багато різноманітних проблем.

Для розв'язання цієї проблеми важливо розуміти наступне: використання касичних методів апроксимації неможливе за двох причин: розглядаєтсья задача апроксимації значень багатовимірної функції, значення якої залежить від вектору параметрів, та класичні методи є занадто грубими та стосуються лише варіантів, коли природа значень тої чи іншої функції відома заздалегідь.

Наприклад, для розв'язання класичної задачі прогнозу певних економічних процесів доцільно використовувати моделі авторегресії через сезонність тих чи інших процесів. Ров'язання задачі прогнозування розвитку певної популяції вимагає використання різницевих рівнянь до дифрівняння, яке дозволяє оцінити параметри системи і за природою оцінити майбутній розвиток популяції. Розглядаючи певні моделі коливань (наприклад, поверхні води) слід

використовувати математичний апарат рівнянь математичної фізики. У ситуації із системним аналізом все набагато складніше.

Тейлор довів, що деяку функцію можна розкласти в околі точки у поліном нескінченної степені, відомий під назвою “ряд Тейлора”. Дійсно, ми маємо право використати такий розклад, модифікувавши його до многовимірного вигляду та оцінити значення коефіцієнтів поліному, наприклад, за допомогою квазіньютонівських методів оптимізації функції похибки, що обрховується як сума квадратів похибок у вузлах. Про такий метод має ряд нюансів: занадто велика кількість змінних до оцінки призводить до низької швидкості збіжності методу в околі точки розв’язку. Такий метод не є ефективним, бо занадто велика кількість параметрів, яку ми маємо оцінювати методами оптимізації.

2.6 Нейронні мережі та універсальна теорема апроксимації

Маючи справу із відображенням $F : X \to Y, X \subset R^n, X \subset R^m$ слід звернути увагу на такий математичиний апарат як нейронні мережі. Звичайний двошаровий перцептрон має наступну структуру (рис 2.2)

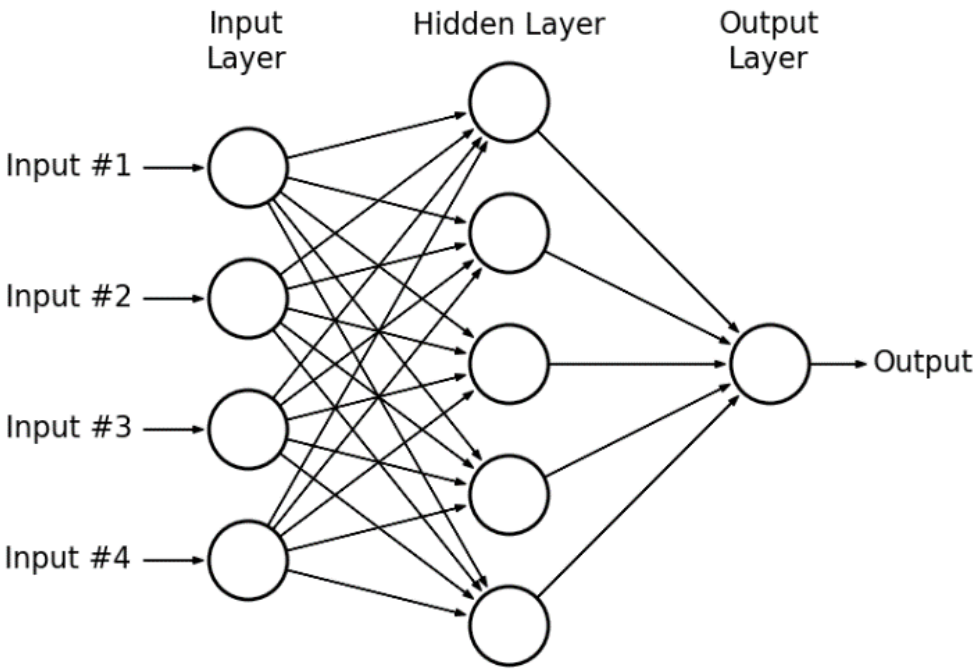


Рисунок 2.2 – Схема перцептрону

На вхід подається певний вектор $X = \{x_1, \ldots, x_N\}$ із вхідних параметрів, які і будуть виступати аргументами. Далі ці значення передають до “прихованого”

шару перцептрона $W = \{W_1, \ldots, W_M\}$, який містить певну множину однотипних нейронів, що виконують функцію суматора. Ці нейрони виконують наступну операцію [3]:

$$W_i(X) = \sum_{j=1}^{N} w_{ij} x_j \tag{2.12}$$

Цю модель можна модифікувати за допомогю додавання “вільного” входу, така модифікація збільшує апроксимуючі та обчислюючі можливості перцептрона. Мова йдеться про значення контанти Θ, яка буде додаватися до вже описаної суми:

$$W_i(X) = \sum_{j=1}^{N} w_{ij} x_j + \Theta_j \tag{2.13}$$

І після цього обраховується значення вихідного шару Y:

$$W_i(X) = \sum_{j=1}^{N} (\sum_{j=1}^{N} w_{ij} x_j + \Theta_j) \tag{2.14}$$

Проте така модель перцептрону є недостатньо точною для виконання якихось складних задач та апроксимацій. Розглянувши таку модель можна побачити, що така функція є звичайною лінійною і особливої точності навідміну від звичайної лінійної регресії набувати не буде.

Проте все змінилось, коли почала йти мова про функції-активатори. Наприклад, класичною функцією можна назвати наступну (одинична ступенька):

$$\varphi(\xi) = \begin{cases} 0, \xi < 0 \\ 1, \xi \geqslant 0 \end{cases} \tag{2.15}$$

Графічно це виглядає наступним чином (рис 2.3)

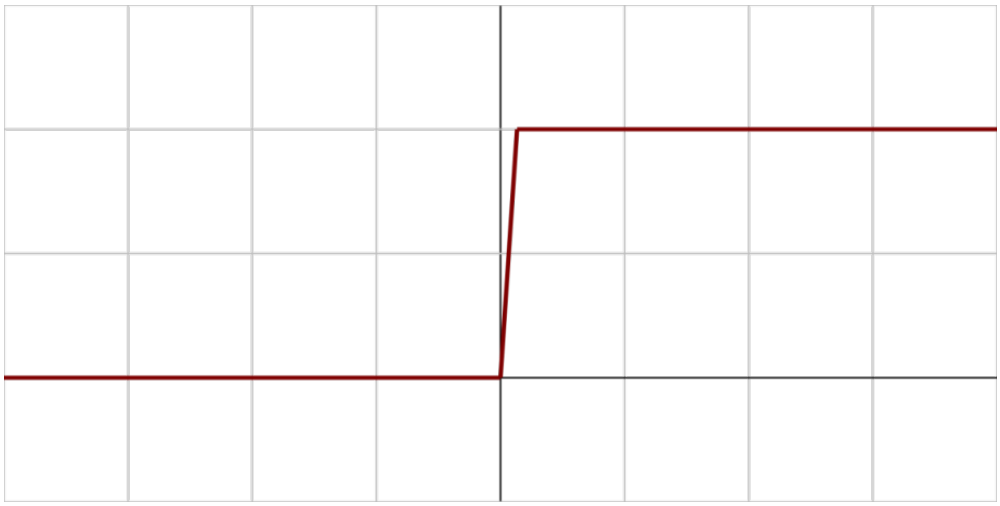


Рисунок 2.3 – Графік одиничної ступеньки

Проте, незважаючи на певну модифікацію звичайного перцептрону, така модель неспроможна видавати необхідні результати, бо за своєю сутністю така нейронная мережа як функція не може точно апроксимувати нелінійні функції [5].

Значно краще використовувати як функції-активатори гладкі функції, які додають до нейронної мережі певну “нелінійність”. Наприклад, “гладка ступенька”:

$$\varphi(\xi) = \frac{1}{1 + e^{-x}} \tag{2.16}$$

Воно має графік (рис 2.4):

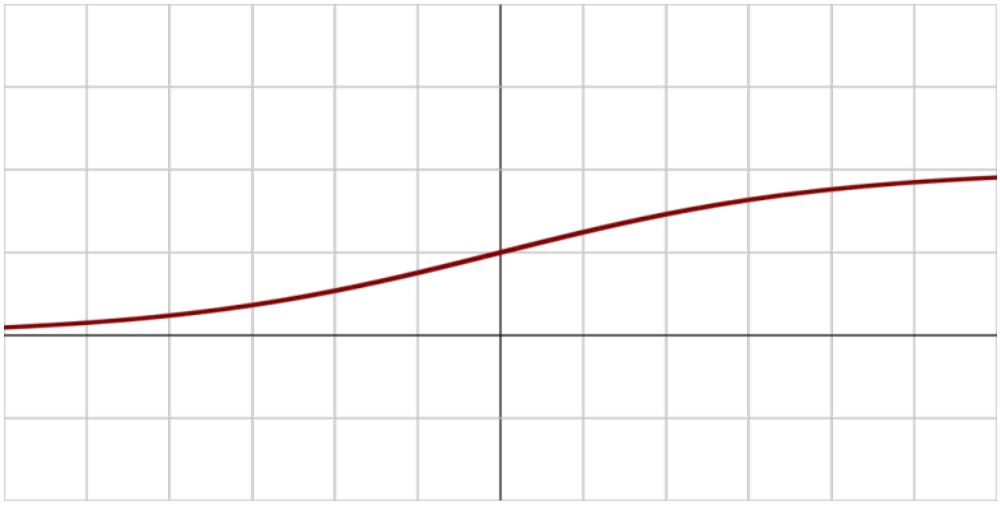


Рисунок 2.4 – Ескіз графіка “гладкої ступеньки”

Така функція має нескінченний порядок гладкості і додавання гладких сигмоїдальних (графік яких схожий на букву S) функції значно покращую апроксимаційні можливості функцій та значно спрощує їх навчання.

Дійсно, якщо мова йде про якусь апроксимацію функцій немає жодної кращої моделі ніж одношаровий персептрон, ефективність якого показує теорема Джорджа Цибенко від 1989 року, відома під назвою Універсальна теорема апроксимації [2].

Нехай задано деяку сигмоідальну функцію, наприклад $\varphi(\xi) = \frac{1}{1+e^{-x}}$. Тоді якщо $f : X \to R$, ($X \subset R^n$- будь яка компактна множина) будь-яка неперервна функція дійсних аргументів, то для будь якої точності $\varepsilon > 0$ існують вектори $w = w_1, \ldots, w_N$, $\alpha = \alpha_1, \ldots, \alpha_N$, $\Theta = \Theta_1, \ldots, \Theta_N$ та параметризована функція $G(*, w, \alpha, \Theta) : X \to R$ така, що:

$$\forall x \in X : |G(x, w, \alpha, \Theta) - f(x)| < \varepsilon \tag{2.17}$$

де $G(x, w, \alpha, \Theta) = \sum_{i=0}^{N} \alpha_i \varphi(w_i x + \Theta_i), w_i \in R_n, \alpha, \Theta \in R^N$.

Фактично ця теорема стверджує про те, що будь-яку функцію можна апроксимувати із заданою точністю за допомогою одношарового персептрону, якщо задати достаню кількість прихованих нейронів.

Цей висновок має ключове значення для вирішення задачі узгодження множини внутрішніх параметрів із множиною значень системи. Ця теорема стверджує, що за будь якої розмірності вектора внутрішніх параметрів існує апроксимуюча значення системи функція для будь якої точності. Більше того, існує і обернена функція, яку можна знайти аналогічним чином і створити повний початковий математичних апарат, необхідний задля пошуку такої множини.

Важливе питання полягає у виборі методу навчання нейронної мережі. Оскільки нашою задачею обрано апроксимацію функцій і за основу обрано певну вибірку із вхідних та вихідних даних, доцільно виконувати пошук коефіцієнтів якимись методами оптимізації. Звісно, що одним із найточніших методів є саме ньютонівські методи, в тому числі їх модифікації квазіньютонівські методи,

які працюють за принципом Ньютона, проте не вимагають взяття оберненого якобіана.

Більш того, методи Ньютона мають змогу виконувати оптимізацію функцій багатьох змінних, проте мова йде про вектор параметрів, які необхідно оптимізувати. Від вхідного шару до прихованого маємо $(N_x + 1)N_h$ вагових коефіцієнтів (з врахуванням "вільного" нейрона), а від прихованого до вихідного при одному вихідному нейроні N_h, тому загальна кількість вагових коефіцієнтів сягає кількості $(N_x + 1)N_h$. Вже при 3 вхідних нейронах та 20 нейронах прихованого шару маємо оптимізацію по 100 параметрам. Розглядаючи квазіньютонівські методи ми зіштовхуємось із великою проблемою обчислювальних складностей і похибок машини, тому слід буде використовувати звичайні методи спуску.

За цільову функції зручніше за все брати суму квадратів похибок. Найпростішим у реалізації слід вважати градієнтний спуск, задаючий таку безумовну оптимізацію функції f по аргументу x:

$$f(x_{x+1}) = x_k - \alpha f'(x_k) \tag{2.18}$$

Тут α задає швидкість спуску, його обирають зазвичай або дробленням, або методом золотого перетину [7].

Як правило, у градієнтного спуску є дуже незручна властивість: як і каже назва, цей метод є методом спуску, навідміну від ньютонівських методів. Якщо функція має "ярну" структуру, то такий метод буде працювати дуже погано, оскільки траекторія спуску буде зигзагоподібною, що дуже погано впливає на швидкості оптимізації.

Певною панацеєю градієнтного метода є стохастичний градієнт, суть якого виражається в наступному:

$$f(x_{x+1}) = x_k - \alpha g(x_k, \mu) \tag{2.19}$$

де $g(x_k, \mu)$ — функція, що залежить від випадкового параметру μ така, що її мат сподівання дорівнює градієнту:

$$E(g(x_k, \mu)) = f`(x_k) \tag{2.20}$$

Їдея стохастичного градієнта полягає в тому, що за допомогою певної випадковості градієнту інколи таким чином функція має можливість "проскочувати" яри і не застрявати надовно в околі одної точки. Практика показує, що звичайний градієнт зазвичай зациклюється в околі двох точок і це дуже сильно понижує швидкість навчання.

Однією із варіацій стохастичного градієнту є так званий "ітеративний" стохастичний градієнт, який працює значно швидше класичного градієнту. Для початку формалізуємо задачу класичної оптимізації градієнтним спуском:

$$F(x) \to min, x \in R \tag{2.21}$$

Для такої оптимізації за великої кількості опимізуючих змінних під час використання звичайного градієнта виникає проблема пов'язана із тим, що оптимізація по всім параметрам виконується одночасно і дані про напрямок спуску впродовж довгого часу одна і та сама, що викликає певну дезінформацію. Ідея ітеративного стохастичного спуску полягає в тому, що за один такт оптимізація відбувається лише за одною оптимізуємою змінною, яка зазвичай вибирається випадково. Виконання n операцій займає один і той самий час, проте такий підхід реалізує поступання більш свіжої інформаціїї для кожної наступної "ітерації" і напрямок часткової похідної є більш точним, ніж за використання класичного градієнта:v

$$x_{k+1} = x_k + \alpha_k g(x_k), a_k = argmin(F(x_k + \alpha_k g(x_k)))$$
$$g(x_k) = \left[0, \ldots, \frac{\partial f(x_k)}{\partial x_k}, \ldots, 0\right]^T, P(g_1(x_k)) \neq 0) = \cdots = P(g_n(x_k) \neq 0) \tag{2.22}$$

2.7 “Potato”-метод, або метод відсікання кінців

Розглянемо складну систему $S\left(Y_{Ny}, U_{N_U}, X_{N_X}, [a,b]_{N_t}\right)$, що має функцію F_ε^S апроксимуючу систему S з точністю ε. Нехай така система має множину Парето $\{P_Y^*, P_U^*, P_X^*\}$ побудовану на множині обмежень системи $\{L_Y, L_U, L_X\}$.

Цілком доцільно стверджувати, що:

$$\begin{aligned}
&\forall i = 0, \ldots, N_Y : [a_i^*, b_i^*]_Y \subset [a_i, b_i]_Y \\
&\forall i = 0, \ldots, N_X : [a_i^*, b_i^*]_X \subset [a_i, b_i]_X \\
&\forall i = 0, \ldots, N_U : [a_i^*, b_i^*]_U \subset [a_i, b_i]_U
\end{aligned} \tag{2.23}$$

Тому за початкові значення множини Парето зручно використовувати множину обмежень. Для пошуку множини внутрішніх параметрів P_X^{k+1} і P_U^{k+1} як підможин P_X^k і P_U^k, що задовольняють наступному для P_Y^k:

$$\forall X \in P_X^{k+1}, \forall U \in P_U^{k+1} : U_\varepsilon(F_\varepsilon^S(X,U)) \cap P_Y^k \neq 0 \tag{2.24}$$

Для початку обираємо швидкість пошуку $\nu > 0$. Ця величина буде характеризувати, наскільки значимі шматки будуть відрізатися від кінців інтервалу, поки ми не пройдемо цю ітерацію. Далі варто обрати напрям звуження: для кожного інтервалу з X та U слід обрати кінець, з якого ми будемо “відрізати” множину внутрішніх параметрів. Далі слід виконати розбиття множини внутрішніх параметрів і для кожної точки обрахувати значення стану. Звужувати інтервал слід там, де це має найменший вплив на довжину інтервалів у порівнянні з минулою ітерацією. Так слід робити до тих пір не отримаємо множини P_X^{k+1} та P_U^{k+1} такі, що: $F_\varepsilon^S(P_X^{k+1}, P_U^{k+1}) \subset P_Y^k$.

Два основних виникаючих питання для такого алгоритму наступні: як обрати швидкість пошуку та як визначити, з якої із сторін варто зробити звуження. Якщо питання вибору швидкості впливає виключно на швидкість збіжності алгоритму, то от визначення сторони, з якої буде проводитись звуження, є критичним і може дуже сильно вплинути на результат. Одним з варіантів, як

це можна зробити, це провести дослідження множини “зсередини”. Розглянемо приклад на R^2 (рис **??**).

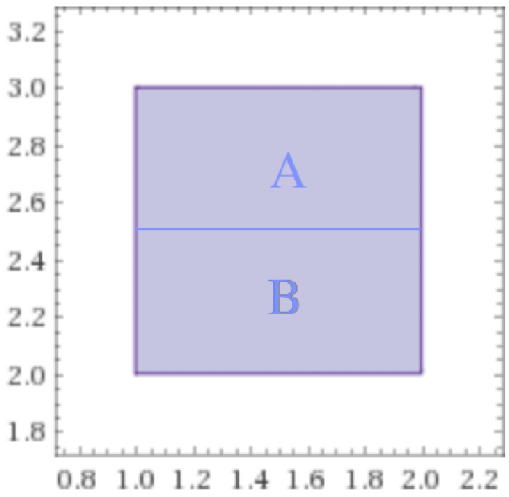


Рисунок 2.5 – Схематичне розділення множини на 2 половини

Наприклад, розглядачи другу координату слід розбити цю множину на дві підмножини. Розгядати всю множину не є доцільним, бо очевидно, що ближче до лінії поділу буде все менше й менше точкок, в яких значення не буде належати множині P_Y^k. Тому має сенс розбити таку множину ще на 2 підмножини:

Дві підмножини A2 та B2, що знаходяться ближче до лінії перетину, не будемо розглядати, а розглянемо лише A1 та B2 (рис 2.6).

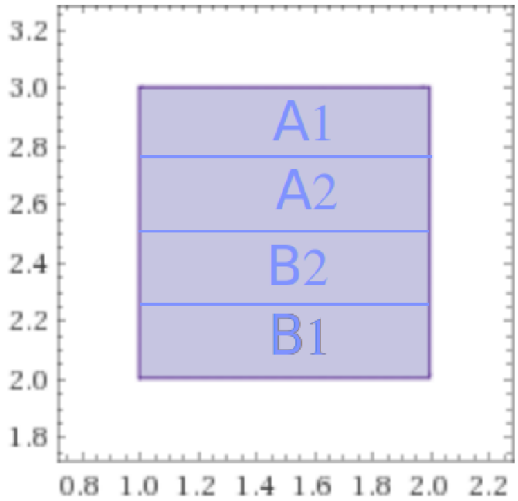


Рисунок 2.6 – Схематичне розділення множини на 4 частини

На етапі аналізу цих двох множин можна легко зіштовхнутись із проблемою вибору кроку дискретизації цих множих для аналізу точок всередині і виявляння

необхідної сторони, з боку якою буде проводитись звуження інтервалу. Якщо крок зробити занадто великим, то інформація буде занадто грубою, а якщо занадто малим, то з таким кроком розрахунки будуть виконуватись дуже довго.

Розбиття з великим кроком рівносильне повному перебору з деякою вірогідністю вибору точки . Наприклад, для розбиття $h_2 = 2h_1$ рівносильне буде за результатом, якщо мати вірогідність врахування точки 0.25, оскільки для пошуку обирається кожна четверта точка, ніж при повному переборі за більшої точності. Оскільки варіант повного перебору є абсолютно неоптимальним, краще використовувати перебір із певною вірогідністю, що точку. буде обрано.

Цілком доцільно вважати, що ближче до центру такого паралелепіпеду (в загальному випадку, коли мова йде про N-вимірний простір) зменшується кількість точок, значення яких не потрапить до не необхідного інтервалу, тому краще робити з більшою ймовірністю пошук на краях паралелепіпеду, ніж у центрі. Ефективно буде використовувати функцію розрахунку ймовірності того, що деяка точка не буде включена (тому що функція із збільшенням відстані зменшує значення, а таку властивість має мати саме ймовірність, що точку буде пропущено, а не обрано) як обернену до відстані між цією точкою та центром паралелепіпеда x:

$$P(y) = \frac{\beta}{d(y,x)} = \frac{\beta}{\sqrt{(y_1 - x_1)^2 + \cdots + (y_n - x_n)^2}} \tag{2.25}$$

Коефіцієнт β залежить від відсотку точок, які ми хочемо перебрати та кроку дискретизації. Його можна обрати наступним чином: якщо задано розбиття T множини Y з певним кроком $\varepsilon > 0$ та середньою вірогідністю вибору точки $\omega > 0$. Скористаємось визначенням першого моменту дискретно заданої випадкової величини:

$$\frac{1}{N}\sum_{y \in T}\frac{\beta}{d(y,x)} = \frac{\beta}{N}\sum_{y \in T}\frac{1}{d(y,x)} = \omega, \beta = \frac{N\omega}{\sum_{y \in T}\frac{1}{d(y,x)}} \tag{2.26}$$

Тепер формалізуємо вибір кінця відрізка, від якого буде відсікатись певна множина точок.

Нехай задано декий швидкість звуження ε множини $([a_1, b_1], \ldots, [a_n, b_n]) = Z \subset R^n$. Нехай для деякої і-ї координати слід звузити таку множину за правилом, яке виражається булевою функцією $B : Z \to \{0, 1\}$, де 0 це неправда, а 1 це правда. Нехай $A_L = ([a_1, b_1], \ldots, [a_i, \frac{3a_i+b_i}{4}], \ldots, [a_n, b_n]), A_L = ([a_1, b_1], \ldots, [\frac{a_i+3b_i}{4}, b_i], \ldots, [a_n, b_n])$. Далі слід утворити розбиття T_L та T_R, що являють собою ε-сітки множин A_L та A_R. Нехай випадкова величина $\xi(z)$ це такая функція як 2.25. Обраховуємо значення $L = \sum_{z \in T_L} \xi(z)B(z), R = \sum_{z \in T_R} \xi(z)B(z)$. Якщо $L > R$, то інтервал будемо звужувати до $([a_1, b_1], \ldots, [a_i + \varepsilon, b_i], \ldots, [a_n, b_n])$, а інакше до $([a_1, b_1], \ldots, [a_i, b_i - \varepsilon], \ldots, [a_n, b_n])$.

2.8 Висновки до розділу

Теоретичне обгрунтування пошуку множини Парето є надзвичайно складною задачею, проте сучасна математично-технічна база має всю необхідну інформацію та машинне забезпечення для того, щоб реалізувати даний алгоритм.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Вибір середовища розробки

Метод відсіканні кінців пошуку множини Парето загалом є надбудовою на brute force (метод “грубої сили”), суть якого полягає в повному переборі можливих параметрів. Навіть з врахуваннями певних модулей мови програмування Python таких як існуючі бібліотеки для машинного навчання TenzorFlow, певні бібліотеки оптимізації та лінійної алгебри такі як SciPy, NumPy, загальна швидкість роботи алгоритму буде дуже сильно страждати, тому що Python навідміну від C-подібних мов програмування є нестрого типізованим і це досягається за допомогою представлення даних у вигляді стрічок тексту, що дуже спрощує обробку даних і роботу з ними, проте жахливо впливає на швидкість обробки даних.

Саме тому для роботи із цим алгоритмом було обрано саме мову програмування C++ із використання фреймворку Qt. Неправдою було б сказати, що такий вибір впав багато завдяки тому, що цією мовою я користувався впродовж довгого часу і цією мовою користуватися найбільш зрудно для мене як загалом для написання будь-якої програми, проте C++ має ще певний ряд корисних властивостей, які вимагає реалізація. По-перше, ця мова програмування комбінує в собі чудовий баланс швидкості та читабельності.

Мова програмування C++ дійсно є однією із найбільш незручних з точки зору читабельності та зрозумілості коду, проте це компенсується гнучкістю мови і можливістю перегрузки операторів, що дозволяє роботи надбудови над існуючими типами даних і перегружати, наприклад, оператори віднімання, множення додавання векторів та матриць, що за оптимальністю може позмагатись із біліотекою NumPy мови програмування Python. Проте розглядаючи поставлене завдання це було б дуже неоптимально і вимагало б великої затрати по часу.

Доцільніше використовувати власну бібліотеку C++, яка дозволяє виконувати роботу із лінійною алгеброю. Ця бібліотека має назву Eigen3. Під час роботи із цією бібліотекою користувач зустрічається із великою кількістю приємних опцій та функцій, що дозволяють зробити розробку легше.

Для початку варто сказати, що серед усіх корисних властивостей цієї бібліотеки слід виділити найбільш цікаву для розробника програмного продукту, а саме недзвичайну швидкість та оптимізацію коду. Квадратну матрицю розмірності в 10 елементів по горизонталі і вертикали Eigen3 виконує на не найбільш сучасному процесорі Intel i7 9th gen не більш ніж за с (час виконання програми в такому випадку сягає кількох сотень наносекунд). Серед істотних мінусів це компіляція, оскільки всі “фішки” бібліотеки Eigen3 зашиті саме в compile time, саме продумана компіляція дозволяю під час виконання отримувати результат настільки швидко.

Інша корисна властивість цієї бібліотеки це її шаблонність. Стандартно бібліотека працює максимально ефективно на вбудованих типад double, який за назвою вдвічі більше за стандартний тип float за розміром використовуємої пам’яті. Якщо останній використовує 4 байти пам’яті, то тип double використовує 8 байтів и може зберігати дані з точністю до 15го знаку після коми у експонентному представленні числа. Проте і на будь яких шаблонних типах даних Eigen3 чудово буде виконувати свої функції. Більше того, за необхідності він може використовувати такі нестандартні типи як std::complex, матриця матриць тощо.

Оскільки мовою програмування було обрано саме C++, для написання програми доцільно використовувати певні надбудови над цією мовою, які дозволяють зручно виконувати чі чи інші задачі по побудові інтерфейсу та зв’язування між собою тих чі інших модулів програми. Багатомісячна практика та неможливість використовувати якісь інші IDE (наприкад, Microsoft Visual Studio C++ не було можливості використовувати через систему Ubuntu) привели до розробки у Qt Creator. Цей фреймворк мови програмування C++ є надзвичайно зручним з точки зору написання певного продукту, який має містити деякий інтерфейс.

Найбільш зручним в Qt Creator можна вважати Qt Designer. Цей інсртумент Qt Creator по своїй сутності є “конструктором” різноманітних елементів інтерфейсу, починаючи від кнопок і закінчуючи окремими від програми повідомленнями поверх програми таких як повідомлення про помилки тощо. За допомогою цього дизайнеру було створено наступну форму (рис 3.1, 3.2).

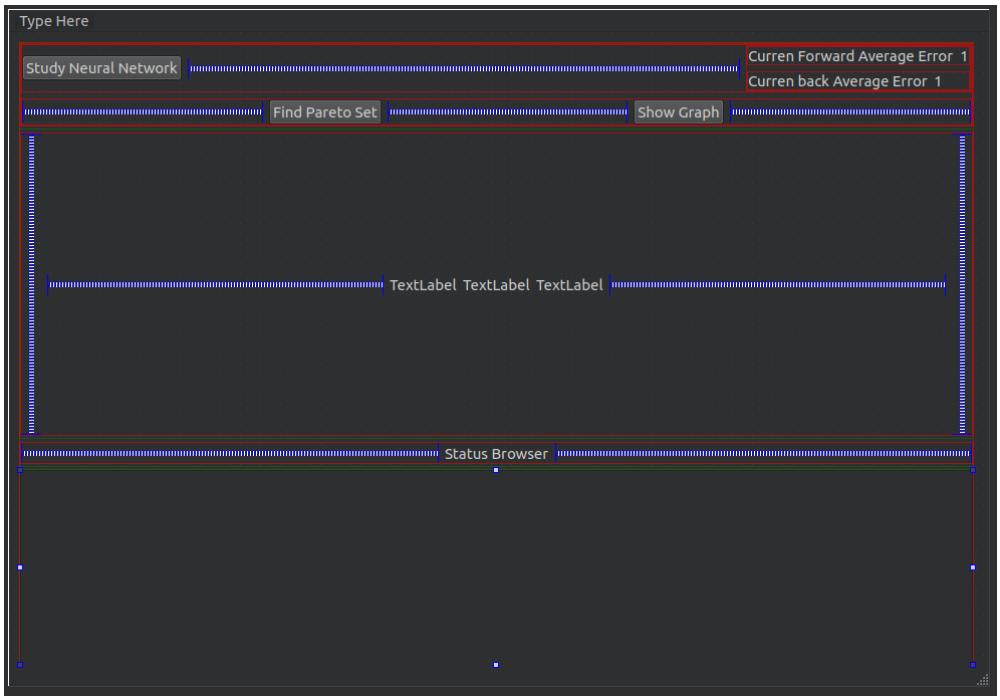


Рисунок 3.1 – Форма програми

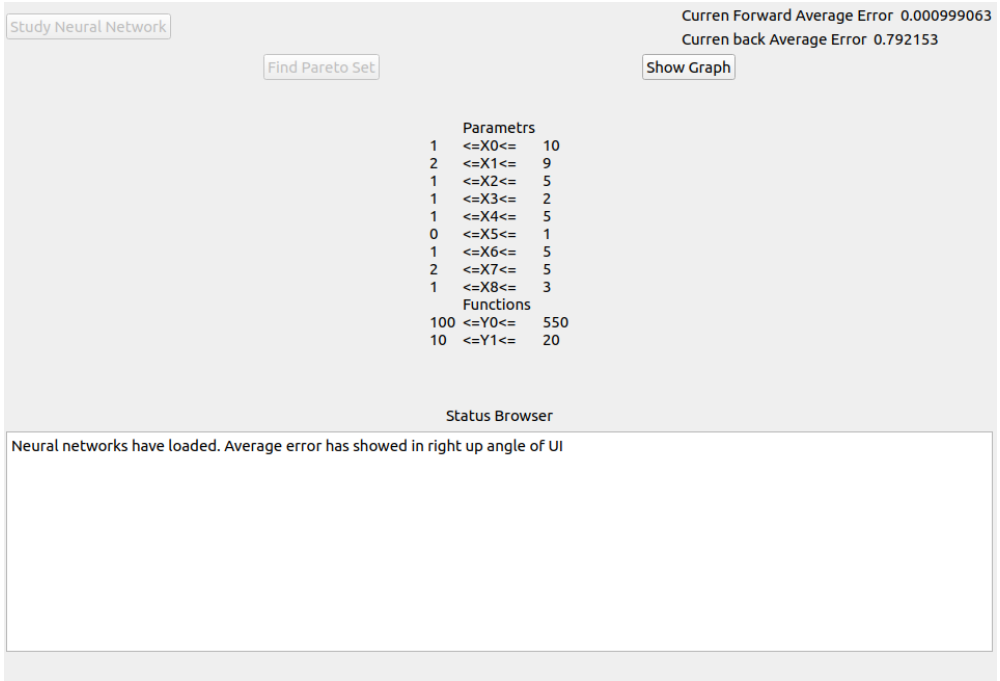


Рисунок 3.2 – Вигляд програми під час запуску

Окрім зручного конструктора форм Qt має концепцію “сигнал-слот”. Будь-який об’єкт може відправляти сигнали іншим об’єктам та приймати сигнали в слоти, виконуючи певний набір команд після прийняття сигналу. Такий мемханізм дозволяє досить зручно працювати із програмою на базовому рівні, тобто

виконуючи певні інструкції після того як користувач натисне на деяку кнопку. Цей механізм реалізується в результаті натиснення правою кнопкою миші на деяку кнопку і вибір необхідного слота. Наприклад, під час створенння програми використовувались слоти “clicked” (рис 3.3.) [8].

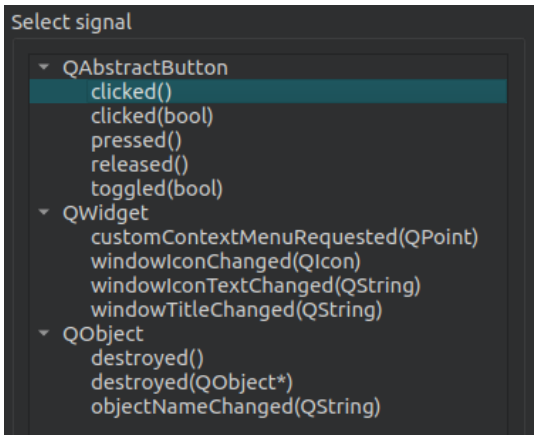


Рисунок 3.3 – Створення зв’язку із сигналом

Вони активуються в результаті “кліку” на кнопку. Існує істотна різниця між “кліком” (сигнал “clicked”) та “натисненням” (сигнал “pressed”). Якщо перший надсилається і сприймається слотом лише після того, як кнопку було відпущено курсором, то останній надсилається одразу як на кнопту було натиснено курсором. Проект виглядає наступним чином (рис 3.4):

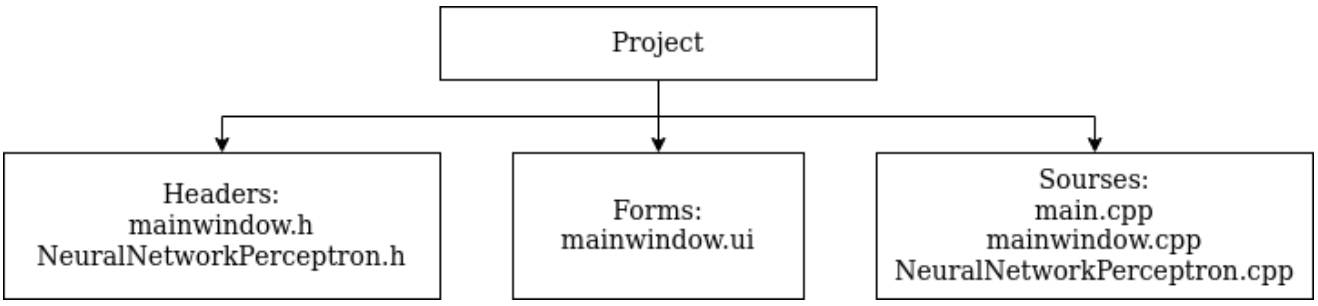


Рисунок 3.4 – Схема проекту

3.2 Реалізація многошарового перцептрону мовою C++

Теорема Цибенка каже, що перцептрону з одним прихованим шаром має бути достатньо для апроксимації будь якої неперервної функції з досить великою точностю , якщо буде достатня кількість нейронів у прихованому шарі. Проте інколи одного шару недостатньо, особливо коли мова йде про деякі неаналітичні

функція. Функціонал має давати можливість створювати перцептрони з якою завгодно кількістю прихованих шарів. Тому ідея реалізації перцептрона наступна (рис 3.5)

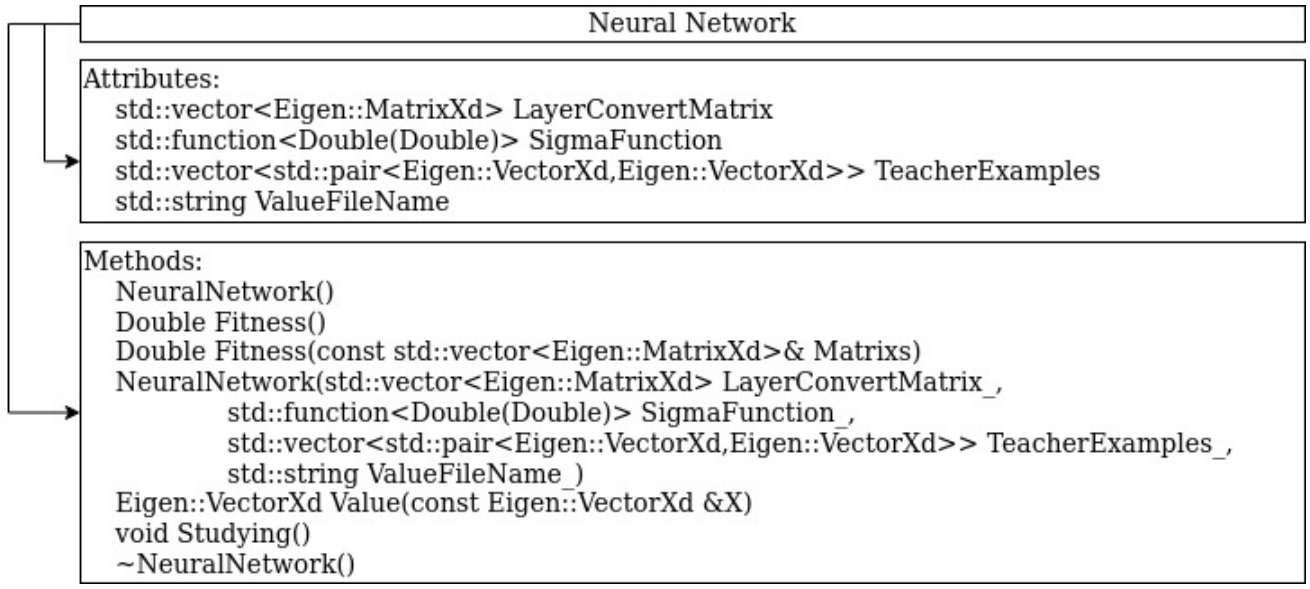


Рисунок 3.5 – Схема класу C++ нейронної мережі

Атрибути класу наступні: вектор матриць Eigen вагів LayerConvertMatrix, сигмоїдальна функція активації SigmaFunction реалізована у вигляді лямбда-функції C++, навчальна вибірка даних TeacherExamples, реалізована у вигляді вектору пар (стандартної бібліотеки C++ std::pair) векторів Eigen, назва файлу, в який буде записуватись набір коефіцієнтів після навчання. Методи класу наступні: стандартні конструктор та деструктор NeuralNetwork та NeuralNetwork, конструктор з параметрами, який є основним методом, за допомогою якого нейронна мережа ініціалізується і вносяться в неї дані, дві функції похибки, реалізовані як обрахунок суми квадратів похибок різниць між між апроксимованими даними за вхідними значеннями навчальної вибірки та навчальними вихідними даними (різниця полягає в тому, що фукнція Fitness без аргументів видає різницю похибок з використанням вбудованих в нейронну мережу вагів, а функція Fitness з параметрами реалізує таку функцію в залежності він конкретного набору вагів). Основними функціями є функції Value та Studying, які використовуються для отримання значення функції за образованими вагами в деякій точці та один крок реалізації. Це виконано для зручної реалізації взаємодії

декілької нейронних мереж в одній і тій самій програмі (щоб навчання могло виконуватись поступово для всій нейронних мереж).

3.3 Опис модулів програми

Головним модулем програми є модуль MainWindow. Він запускається потім із головного файлу main.cpp, проте основним моделем все ж слід вважати MainWindow, оскільки всі дані, змінні, функції тощо зберігаються саме в ньому. Розглянемо його структуру (рис 3.6)

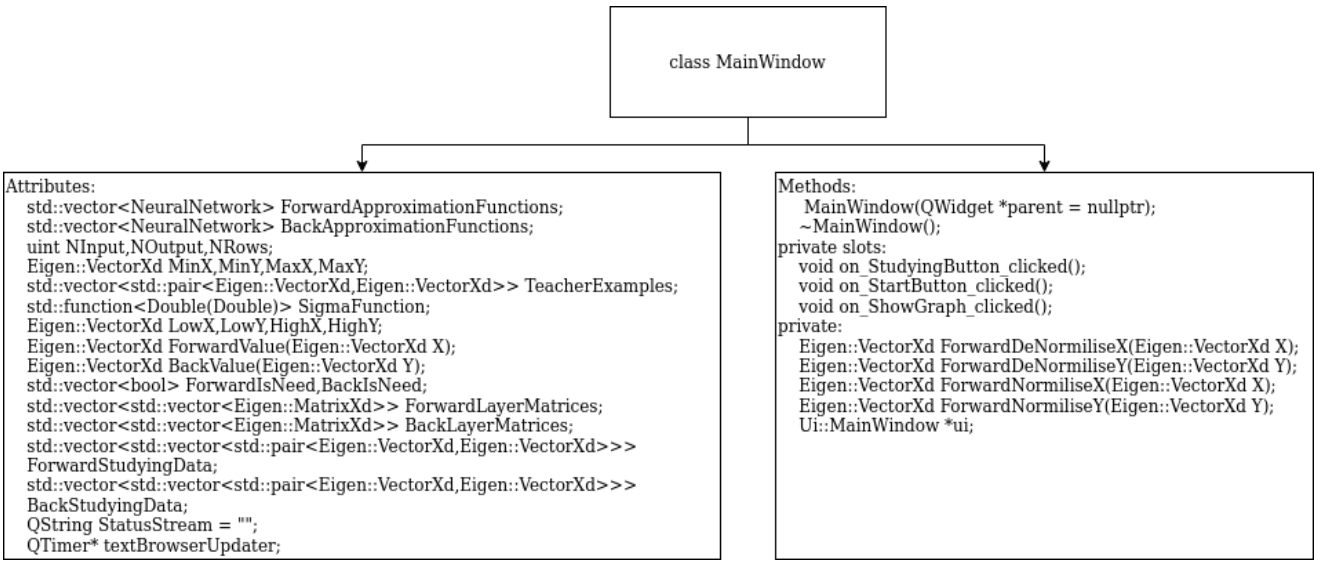


Рисунок 3.6 – Схематичне зображення классу MainWindow

Натискання на кнопку StudyingButton розпочне навчання нейронних мереж, відповідальних за апроксимацію. Оскільки в загальному випадку функцій може бути досить багато, навчання слід виконовувати в декілька потоків. Настикання на кнопку ShowGraph виведе графіки значень навчальних та апроксимованих. Натискання на кнопку StartButton приведе до початку пошуку множини Парето за описаним вище алгоримтом.

3.4 Конкретна задача

У книзі [4] приводить наступна задача. Нехай потрібно за певний термін виконати певну роботу, наприклад, вирити котлован. Роботу можна виконати тільки за допомогою комплексу машин: виїмка землі — екскаваторами, транспортування — вантажівками або тракторами із причепом, укладання вивезеної землі — планувальною машиною (бульдозером, катком тощо). Задача

полягає в тому, щоб підібрати параметри всіх цих машин так, щоб увесь їх комплекс працював ефективно, а роботу було виконано вчасно. Ефективність роботи комплексу залежить від кількох груп параметрів: параметрів котловану (глибини, довжини, ширини, твердості ґрунту), параметрів дороги (довжини і якості), параметрів екскаваторів (місткості ковша), параметрів вантажівок (вантажопідйомності), параметрів бульдозера (потужності) (табл. 3.7).

Номер вибірки	Глибина котловану, м	Довжина котловану, м	Ширина котловану, м	Твердість грунту, число	Довжина дороги, км	Якість дороги, ум.од.	Місткість ковша ескаватора, м^3	Вантажопідйомність вантажівки, т	Потужність, к.с.	Вартість робіт, тис. грн.	Тривалість робіт, год
1	8	8	3	1	3	0	3	2	3	400	18
2	1	4	2	2	5	1	5	3	1	100	14
3	5	5	3	1	3	0	2	4	1	500	12
4	10	8	4	1	5	1	4	5	2	300	20
5	2	7	3	1	5	0	1	2	2	200	10
6	2	6	2	2	3	0	3	4	3	100	11
7	6	3	1	1	3	0	5	3	1	300	17
8	7	5	2	1	5	1	2	2	2	400	19
9	6	4	3	2	1	0	2	5	2	500	13
10	8	4	3	1	4	1	3	2	3	250	10
11	2	3	3	2	1	1	1	3	1	270	14
12	5	5	2	1	2	0	2	2	1	370	17
13	9	8	1	1	5	0	5	5	3	550	19
14	3	7	4	1	3	0	4	4	2	150	15
15	4	6	3	1	2	1	3	3	1	370	13
16	5	5	2	2	1	0	2	2	2	420	18
17	7	2	1	2	1	0	1	4	2	170	11
18	3	9	2	2	5	0	3	3	1	120	10
19	4	7	5	1	4	0	5	2	1	280	17

Рисунок 3.7 – Вибірка даних досліджуваних даних

Дана задача є максимально класичною з точки зору системного аналізу. В залежності від параметрів робіт змінюється вартість робіт та необхідний для цього час. Внутрішні та зовнішні параметри вказані в таблиці (таблиця 1).

Табл. 1 – Внутрішні та зовнішні параметри

<table>
<tr><th colspan="2">Внутрішні параметри</th><th colspan="2">Значення стану системи</th></tr>
<tr><td>X_1</td><td>Глибина котловану, м</td><td rowspan="5">Y_1</td><td rowspan="5">Вартість робіт, тис. грн.</td></tr>
<tr><td>X_2</td><td>Довжина котловану, м</td></tr>
<tr><td>X_3</td><td>Ширина, котловану, м</td></tr>
<tr><td>X_4</td><td>Твердість грунту, число</td></tr>
<tr><td rowspan="2">X_5</td><td rowspan="2">Довжина дороги, км</td></tr>
<tr><td rowspan="5">Y_2</td><td rowspan="5">Тривалість робіт, год</td></tr>
<tr><td>X_6</td><td>Якість дороги, ум.од.</td></tr>
<tr><td>X_7</td><td>Місткість ковша</td></tr>
<tr><td>X_8</td><td>Вантажопідйомність, т</td></tr>
<tr><td>X_9</td><td>Потужність, к.с.</td></tr>
</table>

Вибірка даних складається лише з 19 елементів, тому для більш точних обрахунків її довелося доповнювати додатково. Це не буде давати неправильні дані, якщо у вузлових точках апроксимуюча функція буде давати точні дані. Якщо додаткові елементи будуть дійсно додатковими та не будуть ніяким чином змінювати все існуючі елементи навчальної вибірки даних можна розраховувати на те, що від цього нічого не буде змінюватись. Необхідність вводити деякі додаткові дані полягає в тому, що на 19 точках нейронна мережа видать просто деяку функцію, яка буде давати надпогані дані поза вузлів, в яких проводилась апроксимація.

Цілком логічно, що додавати певні рандомні дані не має ніякого сенсу, тому для стабільної роботи додамо “усереднені” значення вхідних та вихідних параметрів. Наприклад:

$$X_k^* = \frac{X_i + X_j}{2}, Y_k^* = \frac{Y_i + Y_j}{2}, i = 1, \ldots, N, j = 1, \ldots, i-1 \qquad (3.1)$$

де N - розмірність вибірки.

Така операція дозволить додати до вибірки $\frac{N(N-1)}{2}$ елементів, тобто в нашому випадку $\frac{19(19-1)}{2} = \frac{19*18}{2} = 19 * 8 = 171$ та 19 вже існуючих елементів, загалом $171 + 19 = 190$ елементів навчальної вибірки. Слід зазначити, що таке розширення слід виконати строго лише для прямих функцій, для обернених таке розширення не є обов'язковим, оскільки така строга точність важлива лише при проходженні "вперед". Після отримання множини значень внутрішніх параметрів, яка є претендентом на те, щоб стати частиною множини Парето, очікується суттєве зниження міри довжини інтервалів в якому лежить кожний внутрішній параметр.

Це складе суттєві обмеження на допустимі значення обернених функцій і такої великої точності поза фузлами інтерполяції не потрібно, оскільки найбільш важливе та гріздке "відсіювання" буде проведено під час найпершого проходження по множини внутрішніх параметрів.

Швидкість звуження області буде задаватися не константою, а через відсоткове зменшення. По-перше, саме такий вибір кроку зменшення з однієї сторони дозволить на останніх ітераціях не дуже критично зменшувати довжини інтервалів, по-друге, такий інтервал за мірою прямує до нуля: $\lim_{n\to+\infty} c^n = 0, |c| < 1$, тобто:

$$\mu([a_{n+1}, b_{n+1}]_{n+1}) = c\mu([a_n, b_n]_n), |c| < 1$$
$$\lim_{n\to+\infty} \mu([a_n, b_n]_n) = 0$$
$$\lim_{n\to+\infty} (b_n - a_n) = 0 \tag{3.2}$$

Це буде гарантувати отримування інтервалу скільки завгодно малої довжини. За кінцевий критерій закінчення пошуку множини Парето варто взяти зміну по мірі інтервалів, тобто, якщо досліджувана множина характеризується сукупністю інтервалів обмежень $P_n = \{[a_k, b_k]_n\}_{k=1}^N$, кінцем пошуку можна вважати ситуацію, за якої:

$$\sqrt{\sum_{k=1}^{N}(\mu([a_k, b_k]_{n+1}) - \mu([a_k, b_k]_n))} \tag{3.3}$$

3.5 Аналіз результатів роботи програми

Зручніше за все початковими вагами обирати числа близькі до нуля, проте не нуль, оскільки це сповільнить дуже сильно навчання. Через те, що ваги на початку близькі до нуля, сама нейронна мережа видає значення близькі до нуля (рис 3.8):

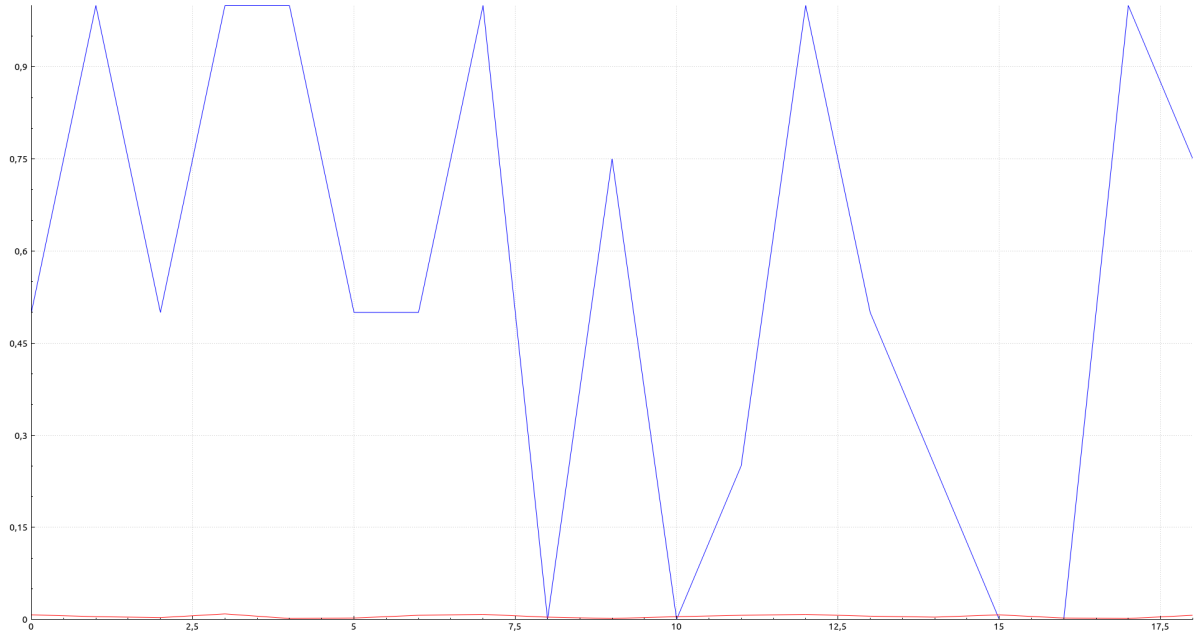


Рисунок 3.8 – Результати апроксимації $f : Y \to X_8$ за стартовими вагами

Завдяки оптимізації градієнтного спуску та грубості стартових значень нейронна мережа має змогу досить швидко вчитися вже на перших секундах. Результати роботи після 5 секунд (рис 3.9):

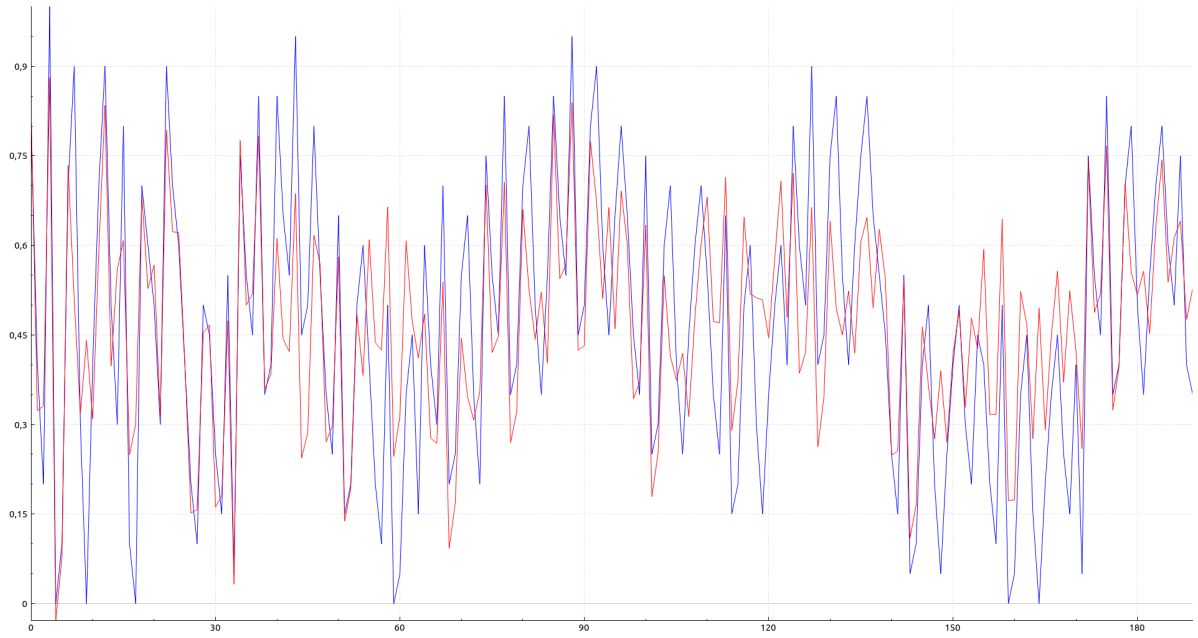


Рисунок 3.9 – Перші ітерації навчання прямої фукції

Вже після першої хвилини роботи вибірка дуже швидко вчиться (рис 3.10).

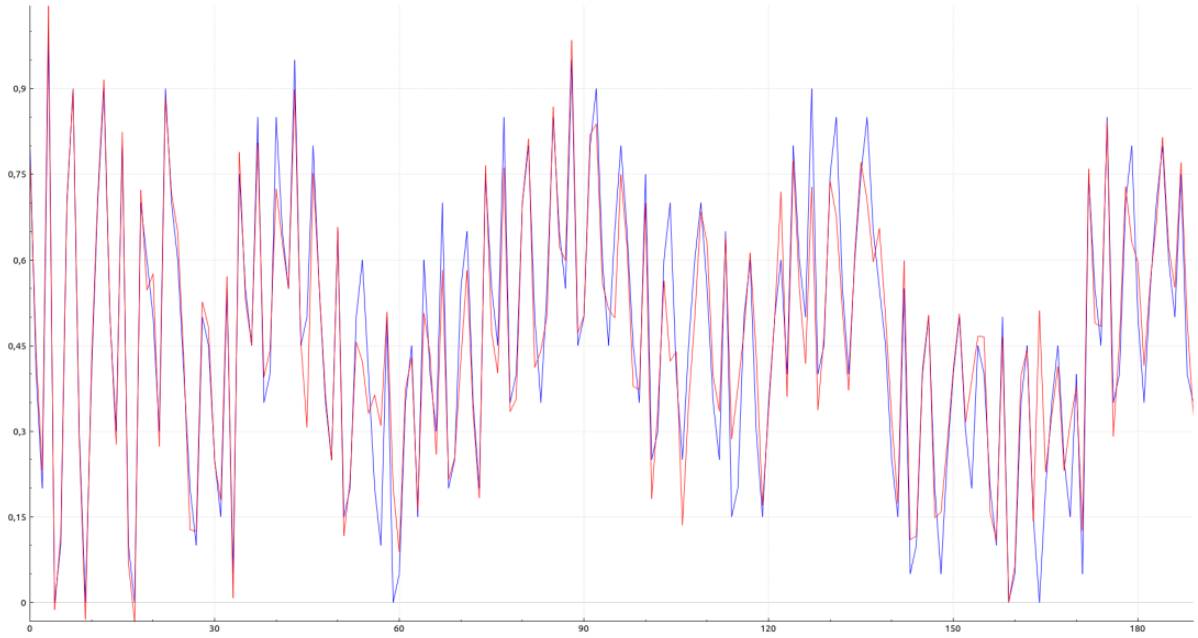


Рисунок 3.10 – Результати роботи після першої хвилини навчання

Обернена функція не має такої точності (рис. 3.11).

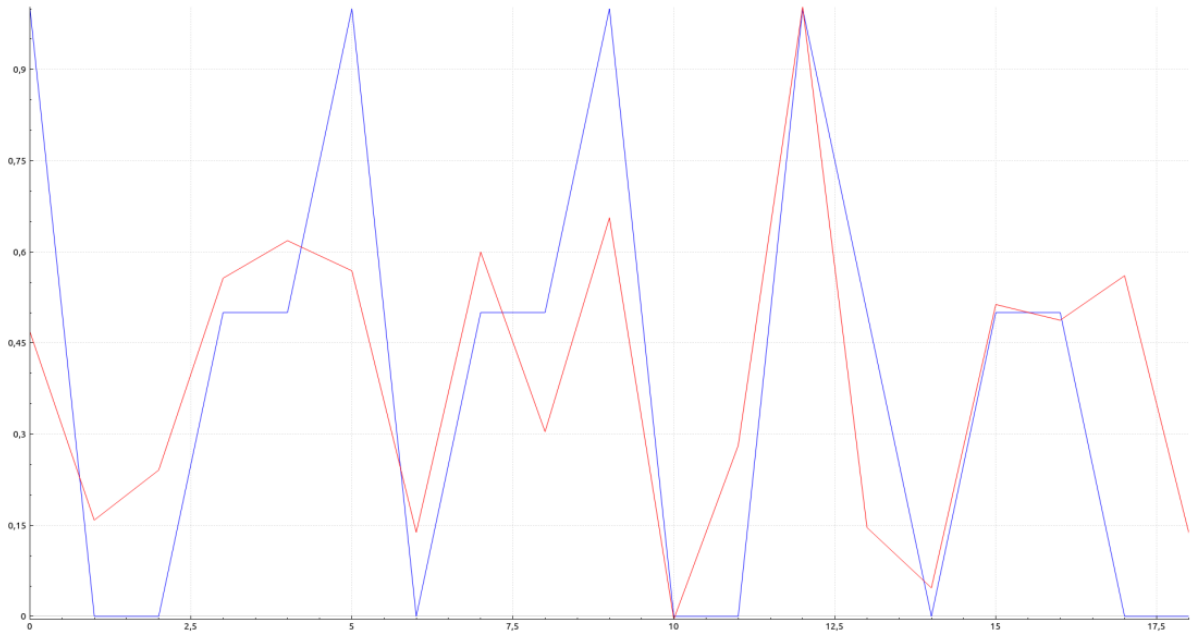


Рисунок 3.11 – Результати навчання обрененої функції після перших ітерації

Навчання обернених функцій була однією із найважчих задач цією роботи. По-перше, навідміну від прямих функцій, їх виділяє абсолютна на перший погляд нелогічність зв’язків з природньої точки зору та більша кількість самих нейронних мереж, по-друге, сама кількість нейроних мереж більша і для економії часу довелось розбивати навчання кожної нейронної мережі на 9 окремик потоків мовою C++ та запускати їх на більш потужному процесорі, який а) підтримує одночасну роботу 12 потоків б) кожен з потоків був у 2 рази потужнішим ніж на пристрої, на якому виконувалась програма. Проте нейронні мережі показали відмінні результати і через 3 години навчання кажна нейронна мережа навчилась до точності в 0.1% (рис 3.12- 3.22):

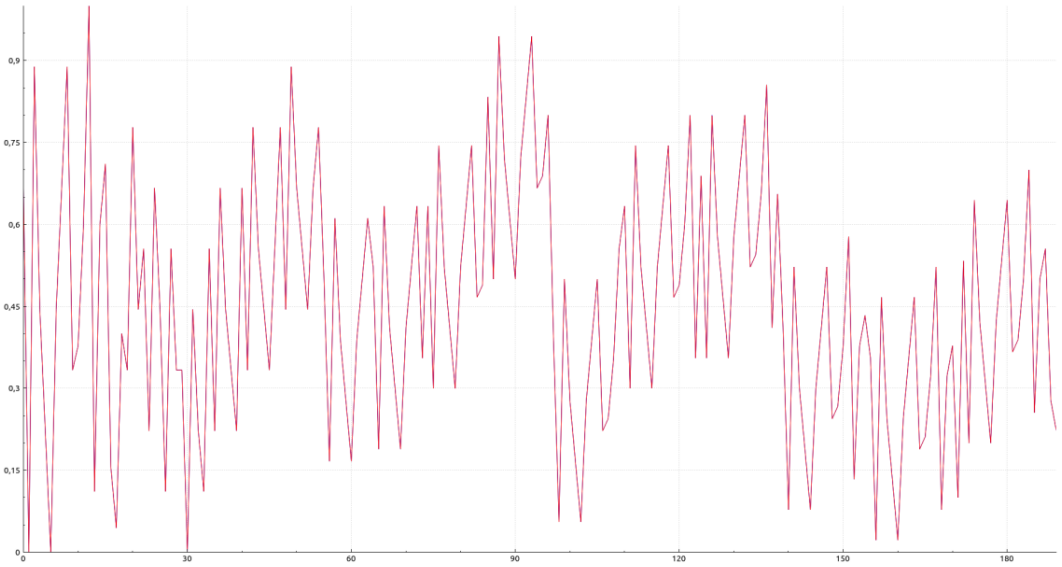


Рисунок 3.12 – Результати навчання функції $f : X \to Y_1$

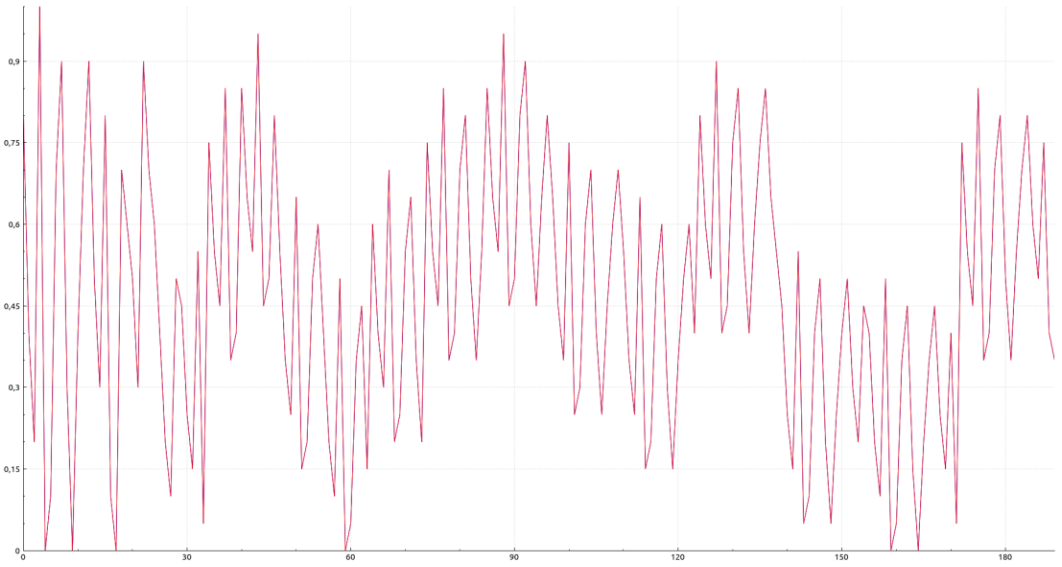


Рисунок 3.13 – Результати навчання функції $f : X \to Y_2$

Рисунок 3.14 – Результати навчання функції $f : Y \to X_1$

Рисунок 3.15 – Результати навчання функції $f : Y \to X_2$

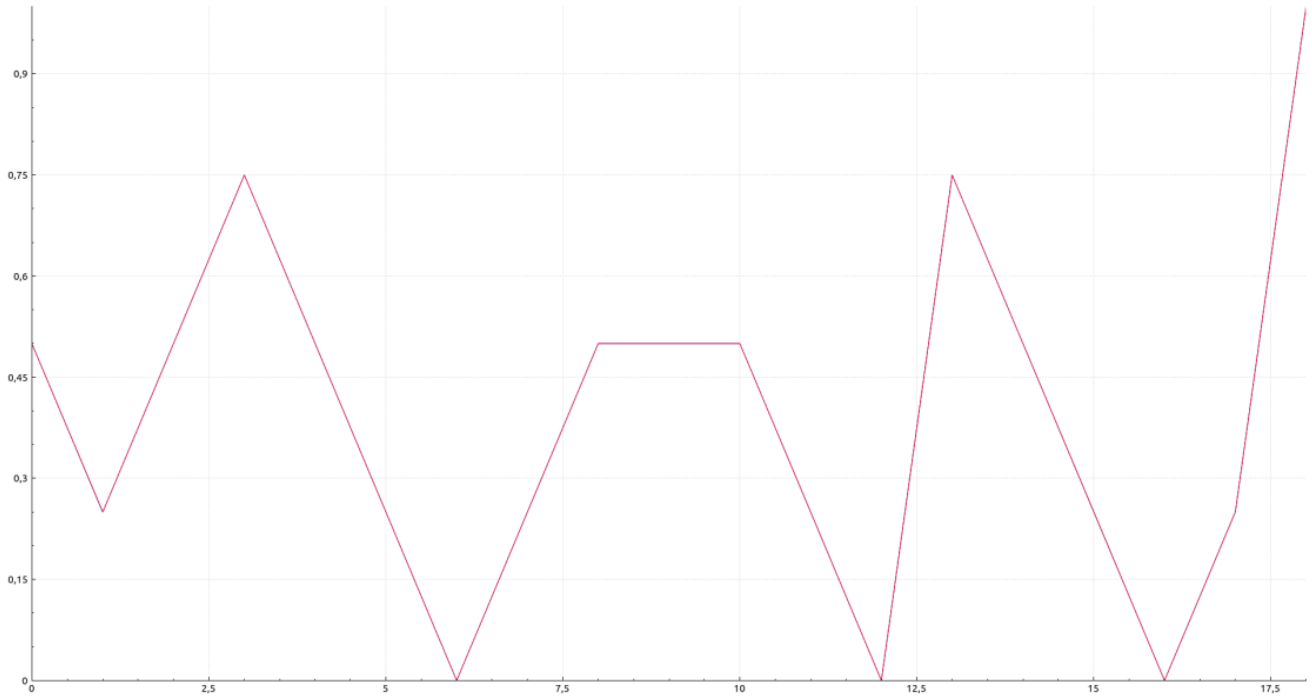


Рисунок 3.16 – Результати навчання функції $f : Y \to X_3$

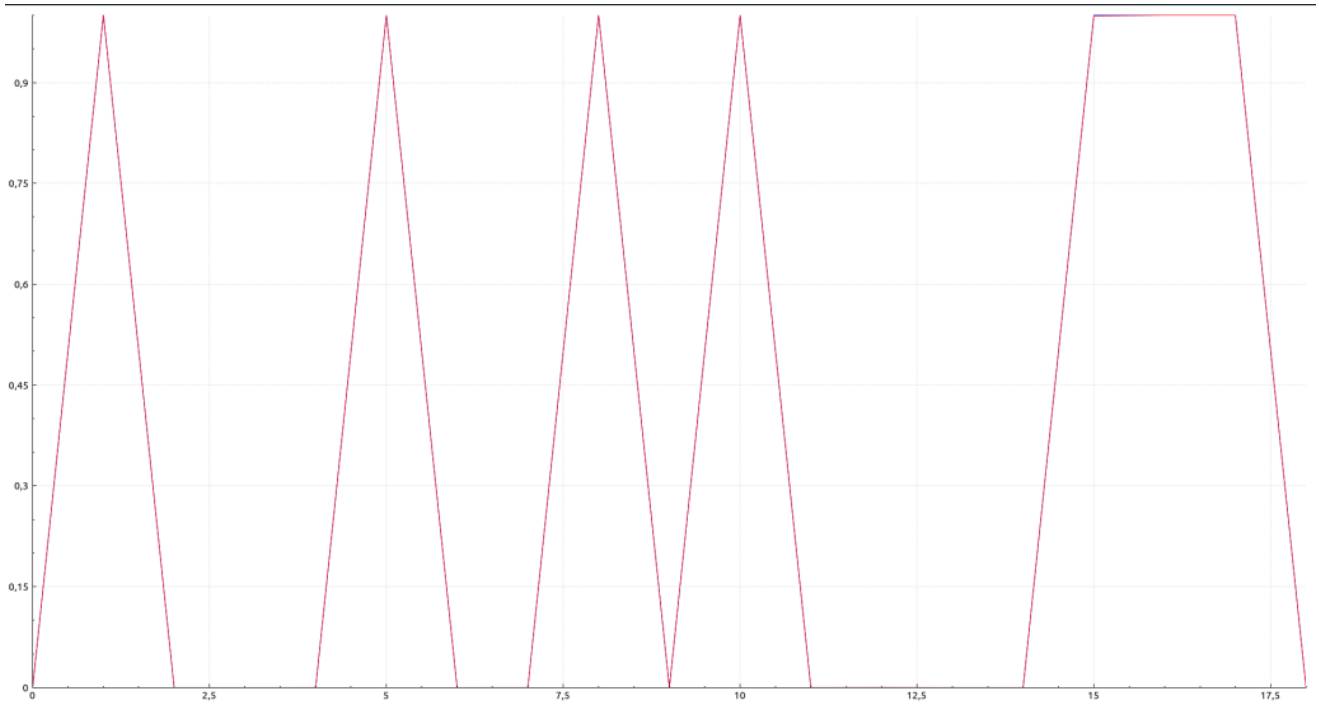


Рисунок 3.17 – Результати навчання функції $f : Y \to X_4$

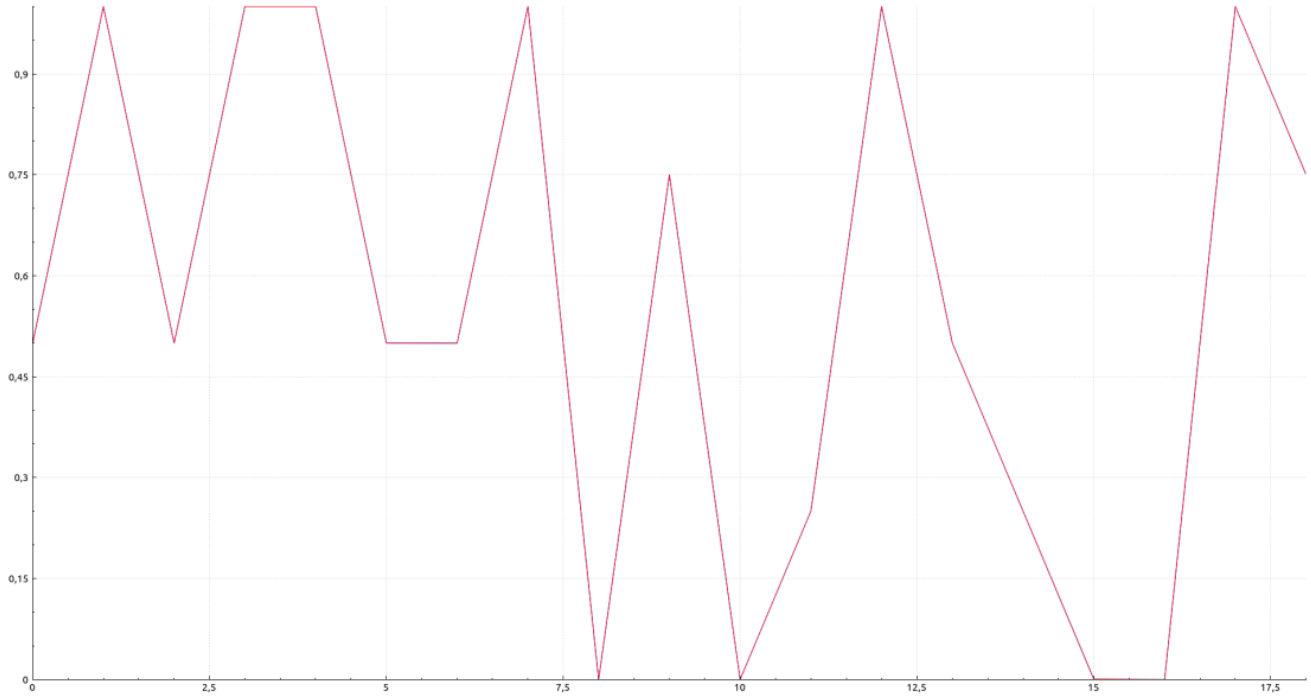


Рисунок 3.18 – Результати навчання функції $f : Y \to X_5$

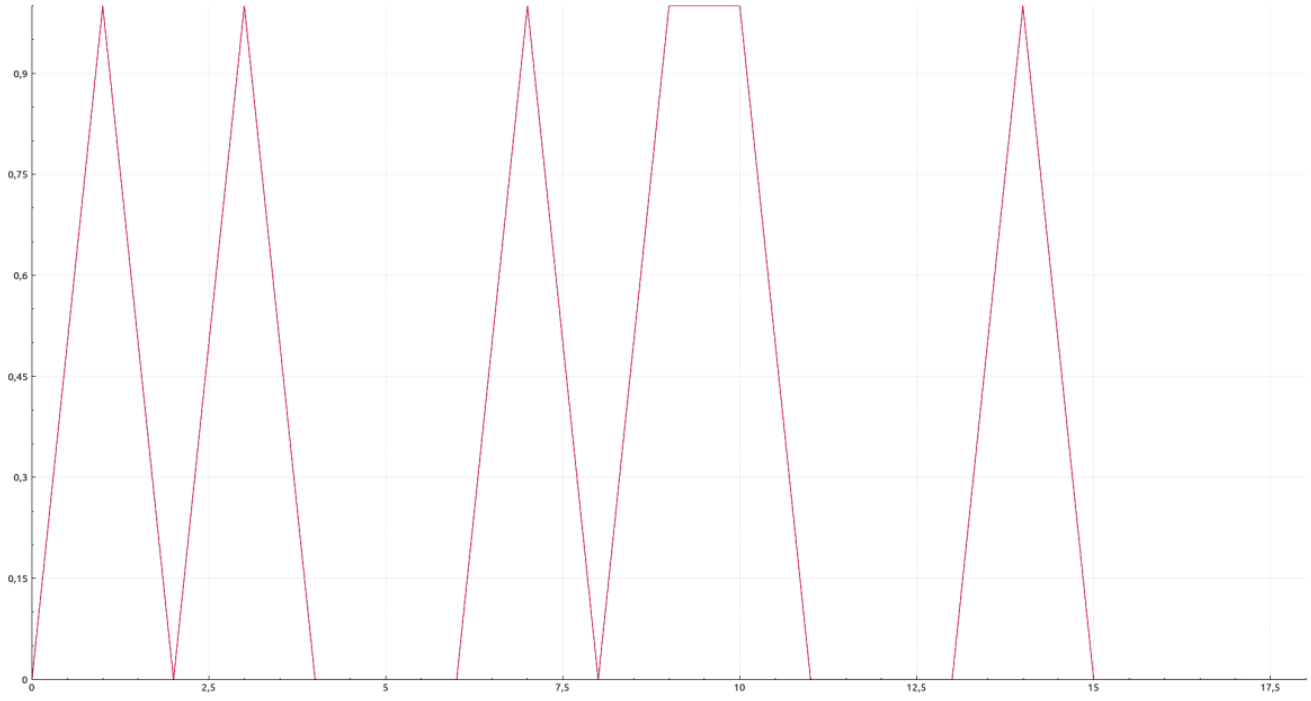


Рисунок 3.19 – Результати навчання функції $f : Y \to X_6$

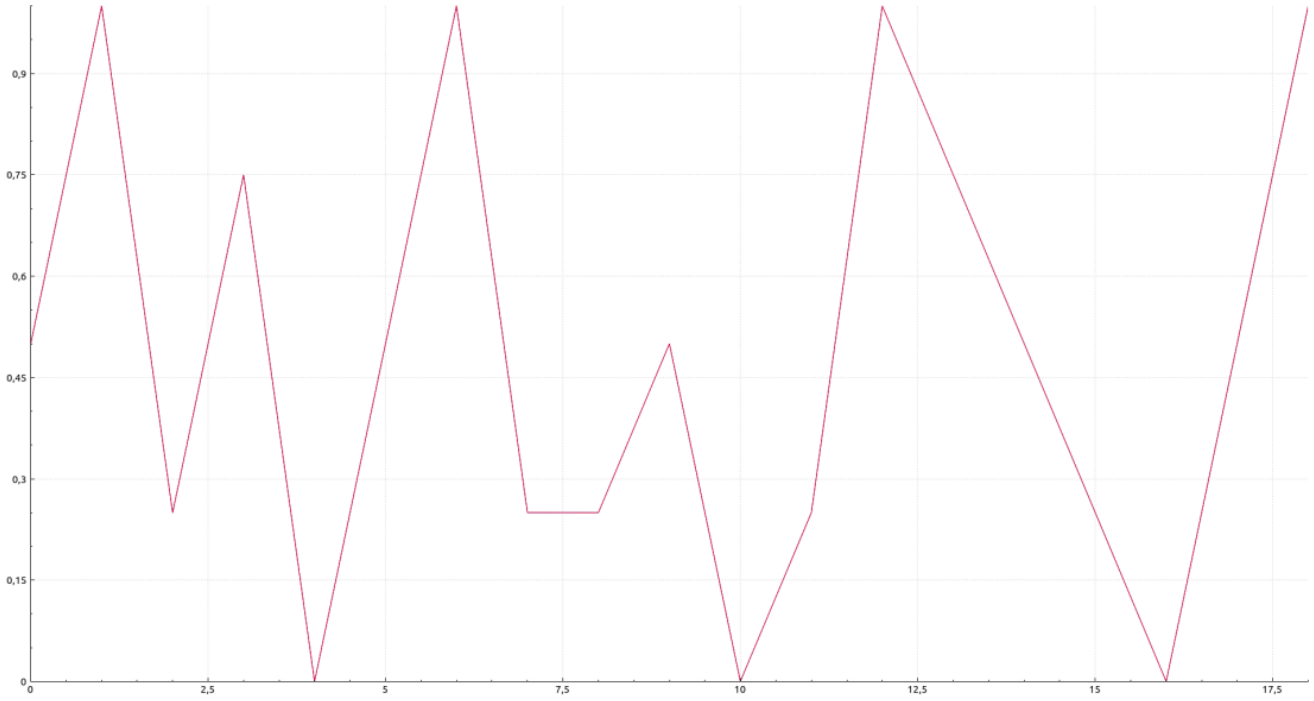


Рисунок 3.20 – Результати навчання функції $f : Y \to X_7$

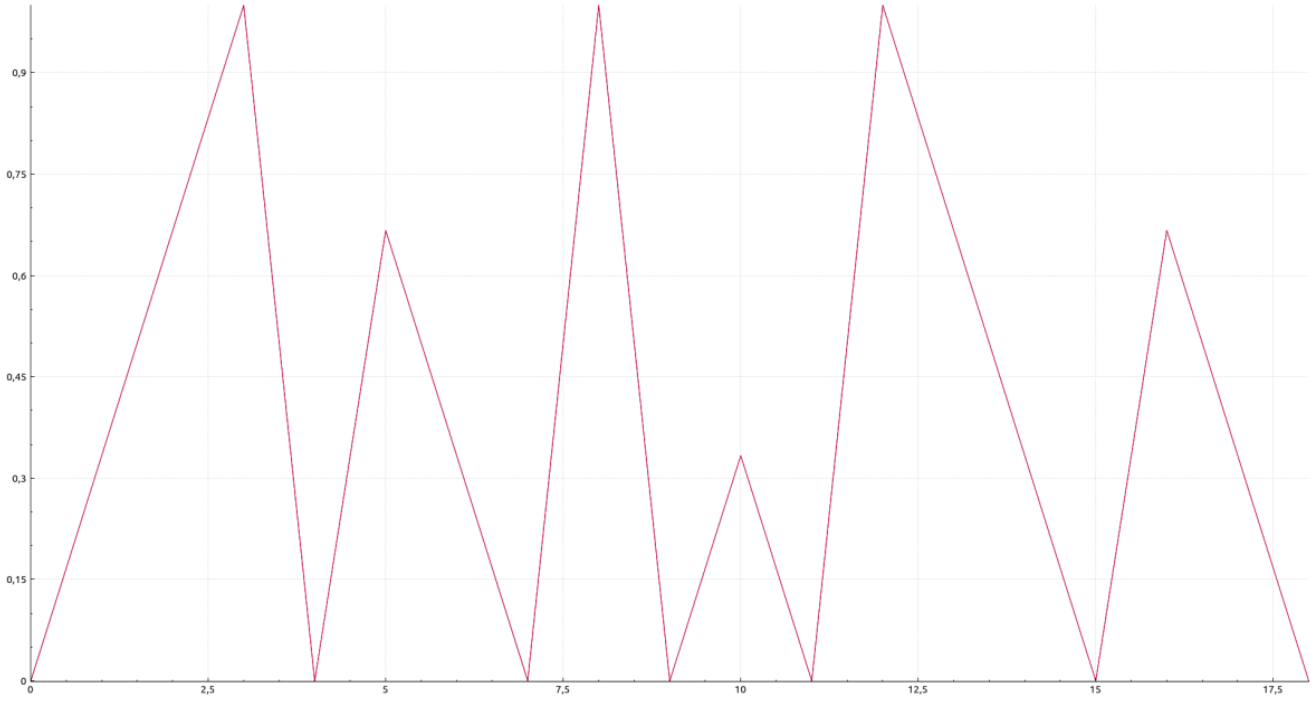


Рисунок 3.21 – Результати навчання функції $f : Y \to X_8$

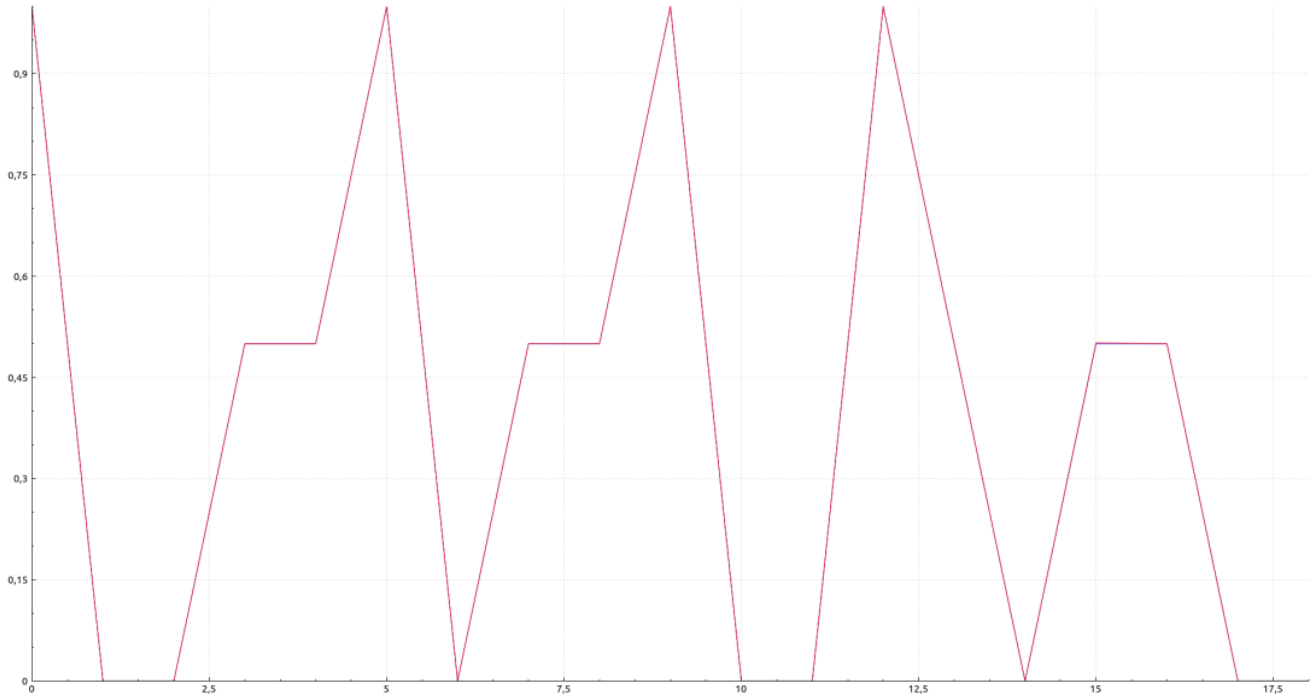


Рисунок 3.22 – Результати навчання функції $f : Y \to X_9$

Виконання звуження проводиться по кожній координаті по черзі. Відповідно до того, з якої сторони програма більшу кулькість непідходящих точок знайшла, з тієї і сторони виконується звуження. За вмовчанням, або зменшити вірогідність випадкового результату, у випадку якщо і зліва і справа однакова кількість непідходящих точок, то звуження виконується справа. Для цих даних звуження відбулось у 2 ієрації: двічі звузилась множина вхідних параметрів та двічі звузилась множина значень системи (рис. 3.23- 3.28):

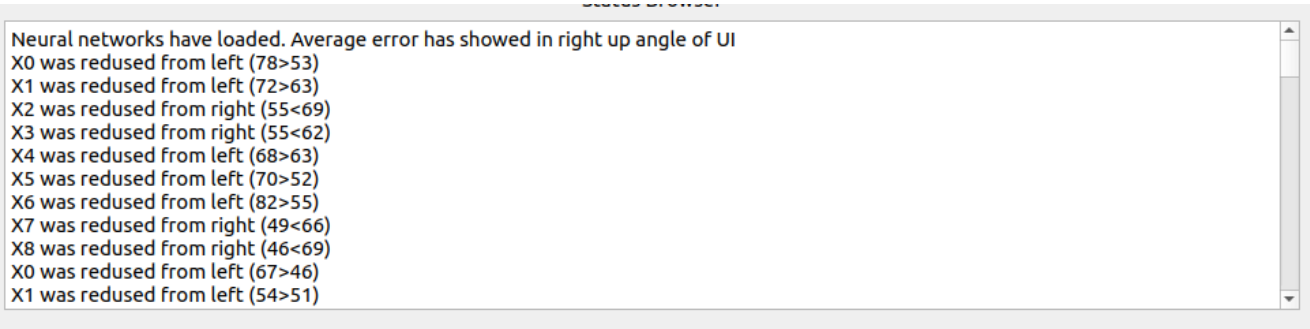


Рисунок 3.23 – Результати звуження множини внутрішніх параметрів на початку першої ітерації

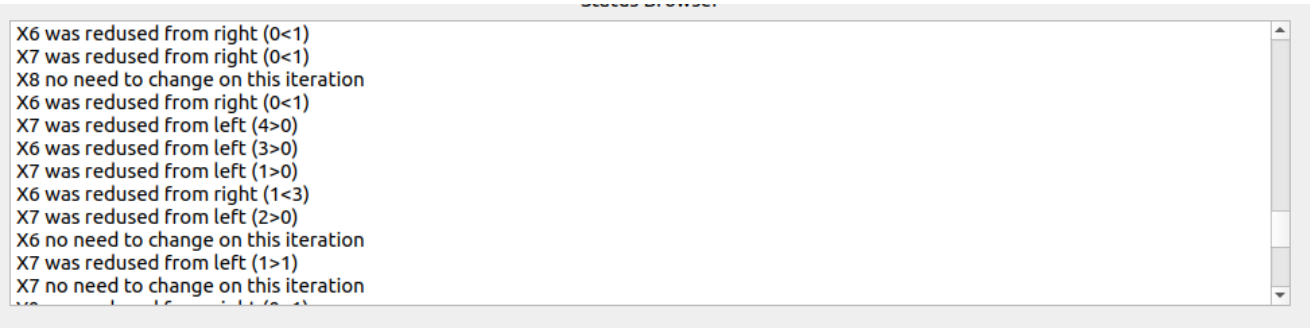


Рисунок 3.24 – Результати звуження множини внутрішніх параметрів в кінці першої ітерації

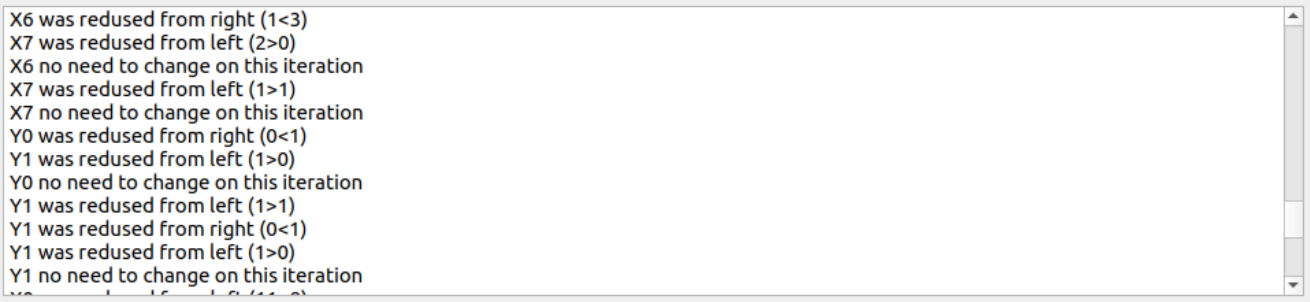


Рисунок 3.25 – Результати роботи по звуженню множини значень системи на першій ітерації

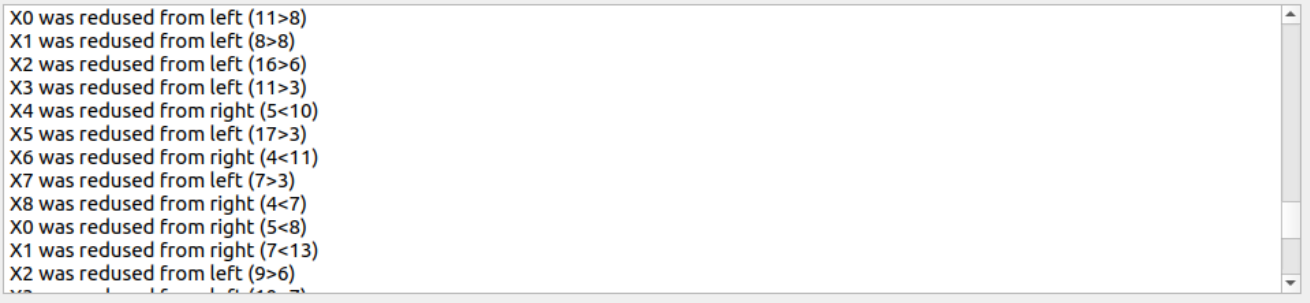


Рисунок 3.26 – Результати звуження множини внутрішніх параметрів в кінці другої ітерації

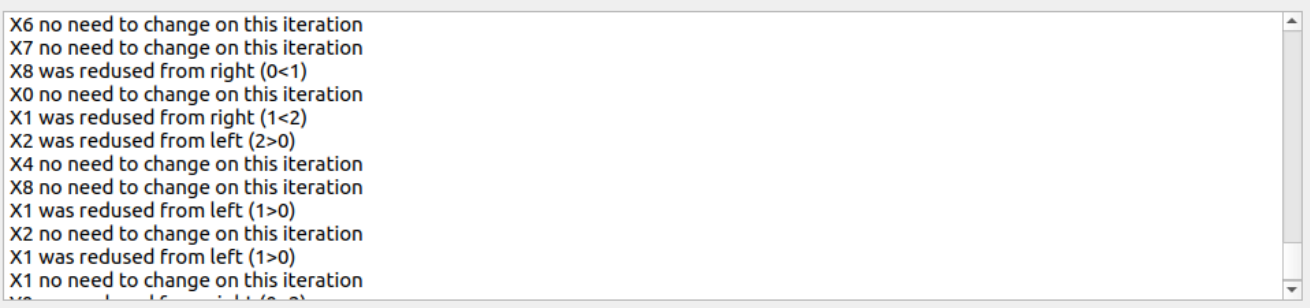


Рисунок 3.27 – Результати звуження множини внутрішніх параметрів в кінці другої ітерації

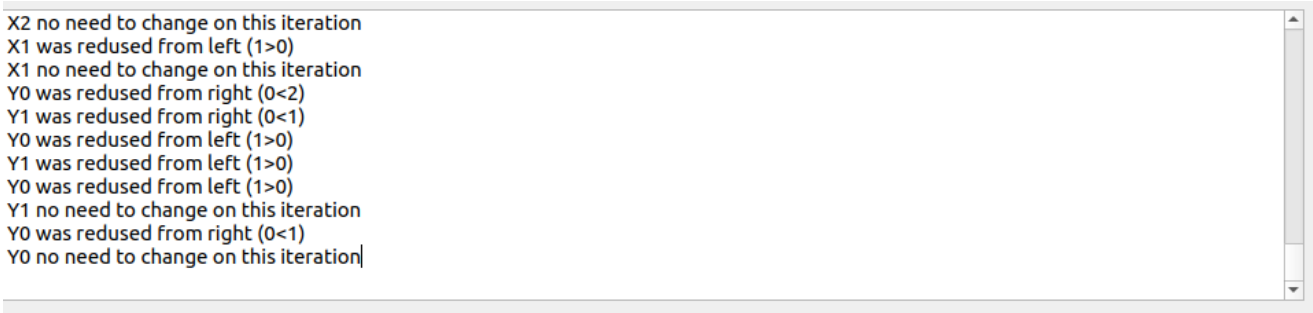


Рисунок 3.28 – Результати роботи по звуженню множини значень системи на другій ітерації

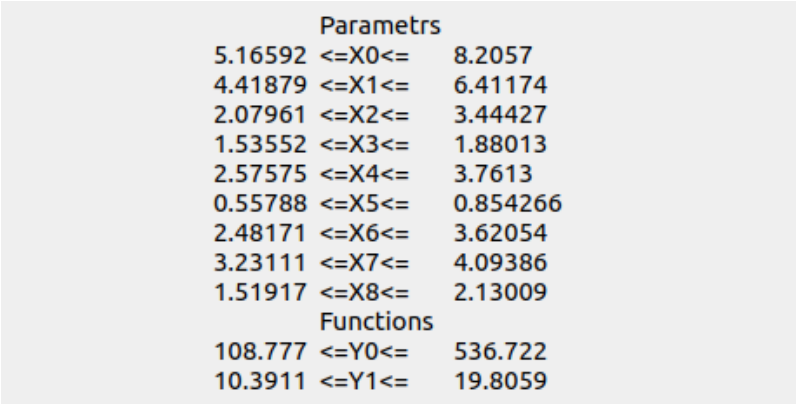


Рисунок 3.29 – Множина Парето, обрахована програмою

Варто помітити, що множина кінцева досить сильно звузилась по внутрішнім параметрам та не дуже сильно по значенням самої системи, це пов’язано з тим що у вибірці дума мало значень щоб надстрого виконувати певні прорахунки “назад”. Сумарний час виконання програми без навчання нейронної мережі складає 30 хвилин.

3.6 Висновки до розділу

Реалізація пошуку множини Парето є надзвичайно складною задачею, проте сучасних математичних знань та обчислювальних потужностей вистачає, аби розв’язати дану задачу.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ

4.1 Постановка задачі

У даному розділі проводиться оцінка програмного продукту призначеного для аналізу та порівняння рекомендаційних систем. Программу було створено використовуючи мову програмування C++ з використанням фреймворку Qt. Середовищем розробки було обрано редактор Qt Creator.

4.2 Обґрунтування функцій та параметрів програмного продукту

F1. - Завантаження навчальної вибірки : а) вибір файлу під час роботи; б) завантаження статичного файлу до початку роботи.

F2. - Навчання апроксимуючої нейронної мережі: а) з вчителем, б) без вчителя.

F3. - Створення алгоритму необхідної точності: а) високої точності, б) низької точності

F4. - Вибір кількості врахованих точок (у відсотках) а) велика кількість, б) низька кількість

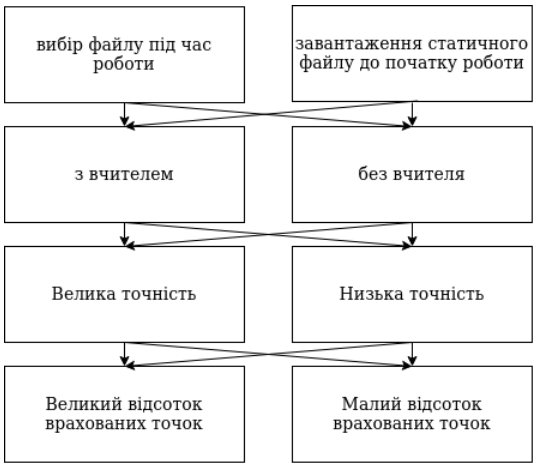


Рисунок 4.1 – Морфологічна карта

Згідно з картою було побудовано позитивно-негативну таблицю (табл ‘2)

Табл. 2 – Позитивно-негативна матриця

Основна функція	Варіант реалізації	Назва	Переваги	Недоліки
F1	А	Вибір файлу під час роботи	Оптимальність та гнучкість	Вимагає втручання ЛПР
	Б	Вибір файлу до початку роботи	Не вимагає втручання ЛПР	Негнучке до незапланованих змін
F2	А	З вчителем	Точніше видає результат	Вимагає втручання ЛПР
	Б	Без вчителя	Не вимагає втручання ЛПР	Видає результат не так точно та повільніше
F3	А	Велика точність	Достовірніший результат	Повільно працює
	Б	Низька точність	Швидко працює	Низька достовірність результату
F4	А	Великий відсоток	Достовірніший результат	Повільно працює
	Б	Малий відсоток	Швидко працює	Низька достовірність результату

За результатами аналізу залишаємо наступні варіанти:

F1A –F2A – F3A - F4A

F1A –F2A – F3A - F4Б

Для оцінки прототипу програмного додатку використовуємо параметри, що будуть описані нижче.

Табл. 3 – Система параметрів додатку

Найменування параметру	Позначення параметру	Значення параметру		
		Мінімальне	Середнє	Максимальне
Час розробки, людина*год	X1	100	157	341
Час роботи алгоритму, с	X2	96	326	781
Точність результату, %	X3	60	73	90
Час, необхідний на перевірку результату, хв	X4	10	17	21

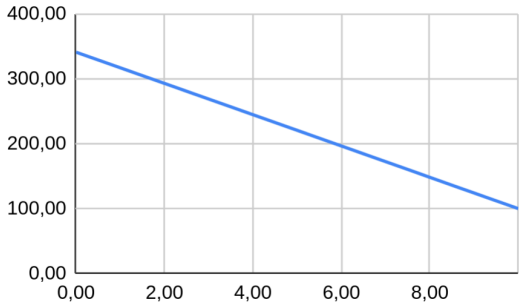


Рисунок 4.2 – Значення X1

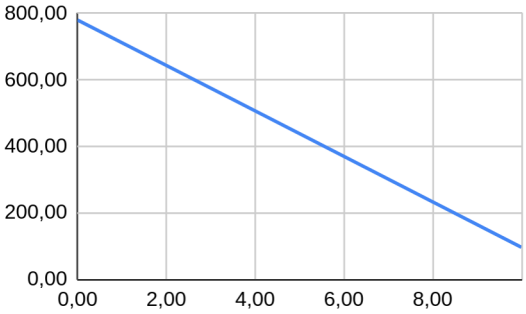


Рисунок 4.3 – Значення X2

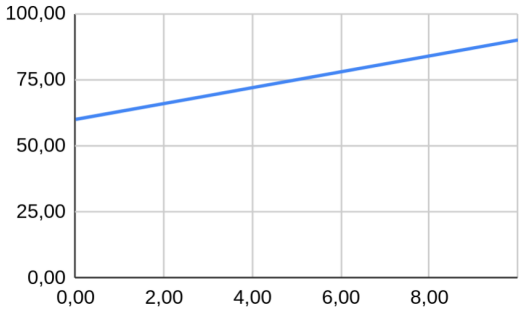


Рисунок 4.4 – Значення X3

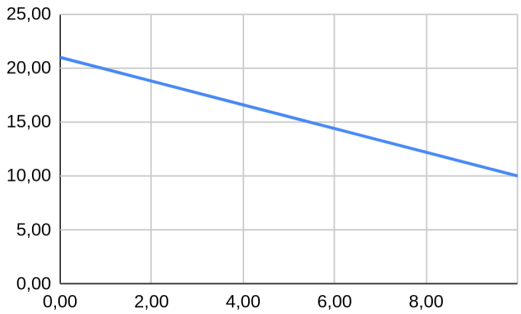


Рисунок 4.5 – Значення X4

Табл. 4 – Результат оцінки параметрів

Параметр	Ранг параметру по оцінці експерта							Сума рангів, Ri	Відхилення Δi	Квадрат відхилення, (Δi)2
	1	2	3	4	5	6	7			
X1	2	2	2	2	3	2	3	16	-1,5	2,25
X2	3	4	3	4	2	4	2	22	4,5	20,25
X3	4	3	4	3	4	3	4	25	7,5	56,25
X4	1	1	1	1	1	1	1	7	-10,5	110,25
Разом	10	10	10	10	10	10	10	70	0	189

Ранг 1 приймається за найменший, а ранг 4 – за найвищий.

Табл. 5 – Попарне зрівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 та X2	<	<	<	<	>	<	>	<	0,5
X1 та X3	<	<	<	<	<	<	<	<	0,5
X1 та X4	>	>	>	>	>	>	>	>	1,5
X2 та X3	<	>	<	>	<	>	<	<	0,5
X2 та X4	>	>	>	>	>	>	>	>	1,5
X3 та X4	>	>	>	>	>	>	>	>	1,5

Коефіцієнт узгодженості дорівнює:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 * 189}{49(64 - 4)} = 0.771 > 0.67$$

Табл. 6 – Розрахунок вагомості параметрів

Параметри	Експерти				Перший крок		Другий крок		Третій крок	
	X1	X2	X3	X4	bi	$K_{вi}$	bi	$K_{вi}$	bi	$K_{вi}$
X1	1,0	0,5	0,5	1,5	3,5	0,219	15,25	0,228	62,375	0,225
X2	0,5	1,0	0,5	1,5	3,5	0,219	15,25	0,228	62,375	0,225
X3	0,5	0,5	1,0	1,5	3,5	0,219	15,25	0,228	62,375	0,225
X4	1,5	1,5	1,5	1,0	5,5	0,344	21,25	0,317	89,875	0,324
Загалом					16	1,000	67	1,000	277,0	1,000

Табл. 7 – Розрахунок коефіцієнтів якості параметрів

Основна функція	Варіант реалізації	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт якості
F1	А	X1	213,000	5,311	0,225	1,196
F2	А	X2	353,000	6,248	0,225	1,407
F3	А	X3	78,000	6,000	0,225	1,351
F4	А	X4	18,000	2,727	0,324	0,885
	Б	X4	17,000	3,636	0,324	1,180

$$K1 = 1.196 + 1.407 + 1.351 + 0.855 = 4.839$$

$$K2 = 1.196 + 1.407 + 1.351 + 1.182 = 5.134$$

Оскільки варіант 2 має більший коефіцієнт якості, він є кращим.

4.3 Економічний аналіз варіантів розробки

1.Завантаження навчальної вибірки під час роботи

2.Навчання апроксимуючої нейронної мережі з вчителем

3.Створення алгоритму високої точності

4.А) Вибір великого відсотку врахованих точок

4.Б) Вибір малого відсотку врахованих точок

Для завдання 1 маємо: алгоритм групи складності 3, ступінь новизни Г, вид використаної інформації БД.

$$T_{\text{р}} = 8, K_{\text{п}} = 0.3, K_{\text{ск}} = 1.07, K_{\text{стм}} = 1$$

$$T_1 = 8 * 0.3 * 1.16 * 1.08 = 2.568 \text{ людино-днів}$$

Для завдання 2 маємо: алгоритм групи складності 1, ступінь новизни Б, вид використаної інформації ПІ.

$$T_{\text{р}} = 43, K_{\text{п}} = 2.02, K_{\text{ск}} = 1.16, K_{\text{стм}} = 1.08$$

$$T_2 = 43 * 2.02 * 1.16 * 1.08 = 168.818 \text{ людино-днів}$$

Для завдання 3 маємо: алгоритм групи складності 1, ступінь новизни А, вид використаної інформації НДІ.

$$T_{\text{р}} = 90, K_{\text{п}} = 1.7, K_{\text{ск}} = 1.16, K_{\text{стм}} = 1$$

$$T_3 = 90 * 1.7 * 1.16 * 1 = 177.48 \text{ людино-днів}$$

Для завдання 4(А) маємо: алгоритм групи складності 3, ступінь новизни Б, вид використаної інформації БД.

$$T_{\text{р}} = 8, K_{\text{п}} = 0.3, K_{\text{ск}} = 1.07, K_{\text{стм}} = 1$$

$$T_{4\text{а}} = 8 * 0.3 * *1.07 * 1 = 2.593 \text{ людино-днів}$$

Для завдання 4(Б) маємо: алгоритм групи складності 2, ступінь новизни Б, вид використаної інформації БД.

$$T_{\text{р}} = 8, K_{\text{п}} = 0.3, K_{\text{ск}} = 1.16, K_{\text{стм}} = 1.008$$

$$T_{4\text{б}} = 8 * 0.3 * 1.16 * 1.08 = 4.992 \text{ людино-днів}$$

Тоді маємо для варіанту А:

$$T_{\text{А}} = (2.568 + 168.818 + 177.48 + 2.593) * 8 = 2811.672 \text{ людино-годин}$$

Тоді маємо для варіанту Б:

$$T_{\text{А}} = (2.568 + 168.818 + 177.48 + 4.992) * 8 = 2830.864 \text{ людино-годин}$$

В розробці бере участь один програміст з окладом 11000 грн та аналітик з окладом 17000 грн.

Розрахунок середньої заробітної плати за годину:

$$С_ч = \frac{28000}{2 * 22 * 8} = 79.54 \text{ грн}$$

Заробітна плата для кожного з варіантів реалізації:

$$С_{зп1} = 79.54 * 2811.672 = 223640/391$$

$$С_{зп2} = 79.54 * 2830.864 = 225166.923$$

Відрахування на соціальне страхування(22%)(грн):

$$С_{від1} = 0.22 * 223640.391 \text{ грн}$$

$$С_{від2} = 0.22 * 225166.923 = 49536.723 \text{ грн}$$

Визначимо витрати котра потрібна на оплату однієї машино-години:

$$С_г = 12 * \ * К_з = 12 * 1100 * 0.25 + 12 * 17000 * 0.25 = 84000 \text{ грн}$$

Враховуючи додаткову заробітну плату:

$$С_{зп} = С_г * (1 + К_з) = 8400 * 1.4 = 117600 \text{ грн}$$

Відрахування на соціалье страхування складатимуть:

$$С_{від} = 0.22 * 117600 = 25872 \text{ грн}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ

– 27000 грн.

$$С_А = К_{тм} * К_а * Ц_{пр} = 1.15 * 0.25 * 27000 = 7763 \text{ грн}$$

Розраховуємо витрати на профілактику та ремонт:

$$С_А = К_{тм} * К_р * Ц_{пр} = 1.15 * 0.05 * 27000 = 1536 \text{ грн}$$

Розраховуємо ефективнийгодинний фонд часу ПК за рік:

$$Т_{ЕФ} = (365 - 104 - 11 - 10) * 8 * 0.9 = 1636 \text{ год}$$

Розраховуємо витрати на електроенергію:

$$С_{ЕЛ} = Т_{ЕФ} * N_С * К_З * Ц_{ЕН} = 1536 * 0.45 * 2 * 1.75 = 2411.213 \text{ грн}$$

Розраховуємо накладні витрати:

$$С_Н = 2700 * 0.67 = 18090$$

$$С_{ЕКС} = 117600 + 25872 + 7763 + 1553 + 2411.213 + 18090 = 173289.217$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = \frac{173289.217}{1536} = 112.818 \text{ грн/год}$$

Враховуючи, що всі роботи ведуться на ЕОМ, витрати на оплату машинного часу:

$$С_{м1} = 112.818 * 2811.672 = 317207.212$$

$$С_{м2} = 112.818 * 2830.864 = 319372.415$$

Накладні витрати відповідно:

$$С_{н1} = 223640.391 * 0.67 = 149839.062$$

$$С_{н2} = 225166.923 * 0.67 = 150861.838$$

Розраховуємо вартість розробки в залежності від варіантів:

$$С_{ПП1} = 223640.391 + 49200.886 + 317207.212 + 149839.062 = 739887.551 \text{ грн}$$

$$С_{ПП2} = 225166.923 + 49536 + 319372.415 + 150861.838 = 744937.899 \text{ грн}$$

Розраховуємо коефіцієнти техніко-економічного рівня для кожного варіанта за формулою:

$$К_{ТЕРj} = \frac{К_{Кj}}{С_{Фj}}$$

Для першого варіанта

$$К_{ТЕР1} = \frac{4.839}{739887.551} = 6.54 * 10^{-6}$$

Для другого варіанта

$$К_{ТЕР2} = \frac{5.134}{744937.899} = 6.892 * 10^{-6}$$

4.4 Висновки до розділу

Отже, можемо зробити висновок, що найбільш ефективним буде варіант з коефіцієнтом техніко-економічного рівня $6.892 * 10^{-6}$, тобто другий варіант. Таким чином, після проведеного функціонально – вартісного аналізу було прийняте

рішення реалізовувати варіант з введенням даних під час роботи програми, навчанням нейронної мережі з вчителем, створення алгоритму високої точності та вибором малого відсотку від досліджуваної множини.

ВИСНОВКИ

Пошук множини Парето є невід’ємною частинох функціонування складних систем, оскільки така множина показує надзвичайно важливі закономірності між множиною внутріш параметрів та множиною значень системи. Нейронні мережі виявились чудовим інструментом для апроксимації функцій, що надзвичайно добре показало себе під час аналізу задачі апроксимації “вперед”. Навіть за умови наявності 190 елементів вибірки нейронна мережа навчилась достатньо швидко і давала досить точні результати, що добре видно під час роботи програми, бо основний час, що програма витрачала на коректування множин, це саме коректування множини значень внутрішніх параметрів. Непоганими з точки зору точності можна вважати кінцеві результати роботи програми та саму множину Парето, яку знайшла програма під час роботи.

Як і очікувалось, такил алгоритм дає досить точні значення множини параметрів. Фактор випадковості у виборі точок може поставити під сумніви результати роботи, проте за законом великих чисел відомо, що середнє арифметичне однаково розподілених випадкових величин дає математичке сподівання величини при достатньо великій кількості розглядаємих значень. Крім того, фактор пов’язаний із швидкістю збіжності такого алгоритму. За півгодини програма змогла знайти множину Парето для системи, що мая 9 вхідних параметрів та 2 вихідних. За необхідності точність такого алгоритму може збільшуватись ще більше за допомогою параллельністю розрахунків, покращенням структури нейронних та більшим часом очікування роботи програми, проте вже на такому на перший погляд недостатньо точному програмному апараті вже за півгодини можна отримати такі точні розрахунки і це не гранична точність роботи такого алгоритму, тому його можна вважати відмінним для такої задачі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Растригин Л. А. Адаптация сложных систем. Рига: Зинатне, 1981. 375 с.

2. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signal, and Systems*. New York, NY: Springer-Verlag New York Inc.,1989, P. 303–314.

3. Бодянський Є.В., Руденко О.Г. Штучні нейронні мережі: архітектури, навчання, застосування. Харків: Телетех, 2004. 369 с.

4. Панкратова Н.Д. Системный анализ: теорія та застосування. Київ: Наукова думка, 2018. 484 с.

5. Шведов А. С. Апроксимація функцій за допомогою нейронних мереж і нечетких систем, *Проблемы управления*. Москва, 2018. Вип. 1, С. 21-29.

6. Лоскутов А. Ю., Михайлов А. С. Основы теории сложных систем. Москва: Ижевськ, 2007. 619 с.

7. Гороховик В.В. Выпуклые и негладкие задачи векторной оптимизаци: Москва: Книжний дом “Либроком”, 2012. 256 с.

8. Шлее. М. Qt 5.10 Профессиональное программирование на C++. Санкт-Петербург: БХВ-Петербург, 2018. 1072 с.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
 1 #include ”mainwindow.h”
 2 #include <QApplication>
 3
 4 int main(int argc, char *argv[])
 5 {
 6     QApplication a(argc, argv);
 7     MainWindow w;
 8     w.show();
 9     return a.exec();
10 }
```

```
 1 #ifndef MAINWINDOW_H
 2 #define MAINWINDOW_H
 3
 4 #include <QMainWindow>
 5 #include <NeuralNetworkPerceptron.h>
 6 #include <QTextStream>
 7 #include <QTimer>
 8
 9 QT_BEGIN_NAMESPACE
10 namespace Ui { class MainWindow; }
11 QT_END_NAMESPACE
12
13 class MainWindow : public QMainWindow
14 {
15     Q_OBJECT
16     std::vector<NeuralNetwork> ForwardApproximationFunctions;
17     std::vector<NeuralNetwork> BackApproximationFunctions;
18     uint NInput,NOutput,NRows;
19     Eigen::VectorXd MinX,MinY,MaxX,MaxY;
20     std::vector<std::pair<Eigen::VectorXd,Eigen::VectorXd>>
          TeacherExamples;
21     std::function<Double(Double)> SigmaFunction;
22     Eigen::VectorXd LowX,LowY,HighX,HighY;
23     Eigen::VectorXd ForwardValue(Eigen::VectorXd X);
24     Eigen::VectorXd BackValue(Eigen::VectorXd Y);
```

```
25      std::vector<bool> ForwardIsNeed,BackIsNeed;
26      std::vector<std::vector<Eigen::MatrixXd>> ForwardLayerMatrices;
          //Matrixces of forward approximation functions
27      std::vector<std::vector<Eigen::MatrixXd>> BackLayerMatrices; //
          Matrixces of forward approximation functions
28      std::vector<std::vector<std::pair<Eigen::VectorXd,Eigen::VectorXd
          >>> ForwardStudyingData; //Studyind examples of forward
          approximation functions
29      std::vector<std::vector<std::pair<Eigen::VectorXd,Eigen::VectorXd
          >>> BackStudyingData; //Studyind examples of back
          approximation functions
30      QString StatusStream = "";
31      QTimer* textBrowserUpdater;
32  public:
33      MainWindow(QWidget *parent = nullptr);
34      ~MainWindow();
35
36  private slots:
37      void on_StudyingButton_clicked();
38      void on_StartButton_clicked();
39      void on_ShowGraph_clicked();
40
41  private:
42      Eigen::VectorXd ForwardDeNormiliseX(Eigen::VectorXd X);
43      Eigen::VectorXd ForwardDeNormiliseY(Eigen::VectorXd Y);
44      Eigen::VectorXd ForwardNormiliseX(Eigen::VectorXd X);
45      Eigen::VectorXd ForwardNormiliseY(Eigen::VectorXd Y);
46      Ui::MainWindow *ui;
47  };
48  #endif // MAINWINDOW_H
```

```
 1  #include "mainwindow.h"
 2  #include "ui_mainwindow.h"
 3  #include <thread>
 4  #include <QCustomPlot>
 5
 6  template <class T1, class T2>
 7  std::pair<T1,T1> operator/(const std::pair<T1,T1> &A, const T2 &B) {
```

```
 8     return std::pair<T1,T1>(A.first/B,A.second/B);
 9 }
10
11 template <class T1, class T2>
12 std::pair<T1,T1> operator*(const std::pair<T1,T1> &A, const T2 &B) {
13     return std::pair<T1,T1>(A.first*B,A.second*B);
14 }
15
16 template <class T>
17 std::pair<T,T> operator+(const std::pair<T,T> &A, const std::pair<T,T
      > &B) {
18     return std::pair<T,T>(A.first+B.first,A.second+B.second);
19 }
20
21 template <class T>
22 std::pair<T,T> operator-(const std::pair<T,T> &A, const std::pair<T,T
      > &B) {
23     return std::pair<T,T>(A.first-B.first,A.second-B.second);
24 }
25
26 MainWindow::MainWindow(QWidget *parent)
27     : QMainWindow(parent)
28     , ui(new Ui::MainWindow)
29 {
30     ui->setupUi(this);
31
32     textBrowserUpdater = new QTimer(this);
33     connect(textBrowserUpdater, &QTimer::timeout, [this]
34     {
35         if(StatusStream.isEmpty()) return;
36         if (StatusStream.back()=='\n')
37             StatusStream.truncate(StatusStream.lastIndexOf(QChar('\n
                  ')));
38         ui->StatusBrowser->appendPlainText(StatusStream);
39         StatusStream.clear();
40     });
41     textBrowserUpdater->start(100);
```

```
42
43      uint N;
44      std::ifstream DataFin(”DataSysan.txt”); //Stream to input
           studying data
45      DataFin >> NRows;
46      DataFin >> NInput;
47      DataFin >> NOutput;
48      ForwardStudyingData = std::vector<std::vector<std::pair<Eigen::
           VectorXd,Eigen::VectorXd>>> (NOutput); //Studyind examples of
           forward approximation functions
49      BackStudyingData = std::vector<std::vector<std::pair<Eigen::
           VectorXd,Eigen::VectorXd>>> (NInput); //Studyind examples of
           back approximation functions
50      for (auto i = 0u; i < NOutput; ++i) { // Get values for all of
           forward functions
51          std::ifstream ForwardLayerFin(”ForwardLayerValue”+std::
              to_string(i)+”.txt”); //Stream for input values
              approximation fnctions
52          ForwardLayerFin >> N;
53          std::vector<uint> ForwardDimentions(N+1);
54          for (auto i = 0u; i < ForwardDimentions.size(); ++i)
55              ForwardLayerFin >> ForwardDimentions[i];
56          std::vector<Eigen::MatrixXd> ForwardLayers;
57          for (auto q = 0u; q < N; ++q) {
58              Eigen::MatrixXd TempLayer(ForwardDimentions[q+1],
                   ForwardDimentions[q]);
59              for (auto i = 0u; i < ForwardDimentions[q+1]; ++i)
60                  for (auto j = 0u; j < ForwardDimentions[q]; ++j)
61                      ForwardLayerFin >> TempLayer(i,j);
62              ForwardLayers.push_back(TempLayer);
63          }
64          ForwardLayerFin.close();
65          ForwardLayerMatrices.push_back(ForwardLayers);
66      }
67      for (auto i = 0u; i < NInput; ++i) { // Get values for all of
           back functions
```

```
68          std::ifstream BackLayerFin("BackLayerValue"+std::to_string(i)
               +".txt"); //Stream for input values approximation fnctions
69          BackLayerFin >> N;
70          std::vector<uint> BackDimentions(N+1);
71          for (auto i = 0u; i < BackDimentions.size(); ++i)
72              BackLayerFin >> BackDimentions[i];
73          std::vector<Eigen::MatrixXd> BackLayers;
74          for (auto q = 0u; q < N; ++q) {
75              Eigen::MatrixXd TempLayer(BackDimentions[q+1],
                   BackDimentions[q]);
76              for (auto i = 0u; i < BackDimentions[q+1]; ++i)
77                  for (auto j = 0u; j < BackDimentions[q]; ++j)
78                      BackLayerFin >> TempLayer(i,j);
79              BackLayers.push_back(TempLayer);
80          }
81          BackLayerFin.close();
82          BackLayerMatrices.push_back(BackLayers);
83      }
84      SigmaFunction = [](Double x){
85          return x/(1+fabs(x));
86      };
87      for (auto i = 0u; i < NRows; ++i) { // Get main studying data
88          Eigen::VectorXd Input(NInput);
89          Eigen::VectorXd Output(NOutput);
90          for (auto j = 0u; j < NInput; ++j)
91              DataFin >> Input(j);
92          for (auto j = 0u; j < NOutput; ++j)
93              DataFin >> Output(j);
94          TeacherExamples.push_back(std::pair<Eigen::VectorXd,Eigen::
               VectorXd>(Input,Output));
95      }
96      DataFin.close();
97      auto LenOfAdd = 2u;
98      for (auto i = 0u; i < NOutput; ++i) {
99          for (auto j = 0u; j < NRows; ++j) {
100             Eigen::VectorXd Input(NInput+1);
101             for (auto q = 0u; q < NInput; ++q)
```

```
102                 Input[q]=TeacherExamples[j].first(q);
103             Input(NInput)=1;
104             Eigen::VectorXd Output(1u);
105             Output[0]=TeacherExamples[j].second(i);
106             ForwardStudyingData[i].push_back(std::pair<Eigen::
                   VectorXd,Eigen::VectorXd>(Input,Output));
107         }
108     }
109     for (auto i = 0u; i < NOutput; ++i) {
110         auto TempData(ForwardStudyingData[i]);
111         for (auto j = 0u; j < TempData.size(); ++j)
112             for (auto k = 0u; k < j; ++k)
113                 for (auto l = 1u; l < LenOfAdd; ++l)
114                     ForwardStudyingData[i].push_back(TempData[j]/
                           LenOfAdd*(LenOfAdd-l)+TempData[k]/LenOfAdd*l);
115     }
116     for (auto i = 0u; i < NInput; ++i) {
117         for (auto j = 0u; j < NRows; ++j) {
118             Eigen::VectorXd Input(NOutput+1);
119             for (auto q = 0u; q < NOutput; ++q)
120                 Input[q]=TeacherExamples[j].second(q);
121             Input(NOutput)=1;
122             Eigen::VectorXd Output(1u);
123             Output[0]=TeacherExamples[j].first(i);
124             BackStudyingData[i].push_back(std::pair<Eigen::VectorXd,
                   Eigen::VectorXd>(Input,Output));
125         }
126     }
127     //    for (auto i = 0u; i < NInput; ++i) {
128     //        auto TempData(BackStudyingData[i]);
129     //        auto H = 0u;
130     //        for (auto j = 0u; j < TempData.size(); ++j)
131     //            for (auto k = 0u; k < j; ++k)
132     //                for (auto l = 1u; l < LenOfAdd; ++l) {
133     //                    if ((++H)%(NRows*(NRows-1)/42)==0) //C++
134     //                        BackStudyingData[i].push_back(TempData[
          j]/LenOfAdd*(LenOfAdd-l)+TempData[k]/LenOfAdd*l);
```

```
135     //                }
136     //    }
137     MinX = Eigen::VectorXd(NInput);
138     MaxX = Eigen::VectorXd(NInput);
139     MinY = Eigen::VectorXd(NOutput);
140     MaxY = Eigen::VectorXd(NOutput);
141
142     for (auto i = 0u; i < NInput; ++i) {
143         MinX[i]=TeacherExamples[0].first(i);
144         MaxX[i]=TeacherExamples[0].first(i);
145         for (auto j = 0u; j < TeacherExamples.size(); ++j) {
146             if (MinX[i]>TeacherExamples[j].first[i])
147                 MinX[i]=TeacherExamples[j].first[i];
148             if (MaxX[i]<TeacherExamples[j].first[i])
149                 MaxX[i]=TeacherExamples[j].first[i];
150         }
151     }
152     for (auto i = 0u; i < NOutput; ++i) {
153         MinY[i]=TeacherExamples[0].second(i);
154         MaxY[i]=TeacherExamples[0].second(i);
155         for (auto j = 0u; j < TeacherExamples.size(); ++j) {
156             if (MinY[i]>TeacherExamples[j].second[i])
157                 MinY[i]=TeacherExamples[j].second[i];
158             if (MaxY[i]<TeacherExamples[j].second[i])
159                 MaxY[i]=TeacherExamples[j].second[i];
160         }
161     }
162     for (auto i = 0u; i < ForwardStudyingData[0].size(); ++i) {
163         for (auto j = 0u; j < NInput; ++j) {
164             for (auto k = 0u; k < NOutput; ++k) {
165                 ForwardStudyingData[k][i].first(j)-=MinX[j];
166                 ForwardStudyingData[k][i].first(j)/=(MaxX[j]-MinX[j])
                        ;
167             }
168         }
169         for (auto j = 0u; j < NOutput; ++j) {
170             ForwardStudyingData[j][i].second(0)-=MinY[j];
```

```
171             ForwardStudyingData[j][i].second(0)/=(MaxY[j]-MinY[j]);
172         }
173     }
174     for (auto i = 0u; i < BackStudyingData[0].size(); ++i) {
175         for (auto j = 0u; j < NInput; ++j) {
176             BackStudyingData[j][i].second(0)-=MinX[j];
177             BackStudyingData[j][i].second(0)/=(MaxX[j]-MinX[j]);
178         }
179         for (auto j = 0u; j < NOutput; ++j) {
180             for (auto k = 0u; k < NInput; ++k) {
181                 BackStudyingData[k][i].first(j)-=MinY[j];
182                 BackStudyingData[k][i].first(j)/=(MaxY[j]-MinY[j]);
183             }
184         }
185     }
186
187     for (auto i = 0u; i < NOutput; ++i) {
188         ForwardApproximationFunctions.push_back(NeuralNetwork(
              ForwardLayerMatrices[i],
189                                                               SigmaFuncti
                                                                  ,
190                                                               ForwardStud
                                                                  [i],
191                                                               "
                                                                  ForwardLa
                                                                  "+
                                                                  std
                                                                  ::
                                                                  to_strin
                                                                  (i))
                                                                  );
192     }
193     for (auto i = 0u; i < NInput; ++i) {
194         BackApproximationFunctions.push_back(NeuralNetwork(
              BackLayerMatrices[i],
195                                                             [](Double
                                                                x){if (
```

```
                                                                x<0)
                                                                return
                                                                0.01*x;
                                                                 return
                                                                 std::
                                                                tanh(x)
                                                                ;},
196                                                     BackStudyingData[i],
197                                                     ”BackLayerValue”+std::
                                                           to_string(i)));
198     }
199
200     LowX = Eigen::VectorXd (NInput);
201     HighX = Eigen::VectorXd(NInput);
202     LowY = Eigen::VectorXd(NOutput);
203     HighY = Eigen::VectorXd(NOutput);
204     for (auto i = 0u; i < NInput; ++i) {
205         LowX[i]=0;
206         HighX[i]=1;
207     }
208     for (auto i = 0u; i < NOutput; ++i) {
209         LowY[i]=0;
210         HighY[i]=1;
211     }
212
213     Double max=0;
214     for (auto i = 0u; i < NOutput; ++i) {
215         auto temp = sqrt(ForwardApproximationFunctions[i].Fitness()/
              ForwardStudyingData.size());
216         if (max<temp)
217             max=temp;
218     }
219     ui->ForwardCurAvErValue->setNum(max);
220     max=0;
221     ui->BackCurAvErValue->setNum(sqrt(SumVecOfNN(
          BackApproximationFunctions)/ForwardStudyingData.size()/NInput)
          );
```

```
222
223     StatusStream +=(QString("Neural networks have loaded. Average
            error has showed in right up angle of UI\n"));
224     QString Left="",Middle="",Right="";
225     Left+="\n";
226     Right+="\n";
227     Middle+="Parametrs\n";
228     for (auto i = 0u; i < NInput; ++i) {
229         Left+=QString::number(MinX[i])+"\n";
230         Right+=QString::number(MaxX[i])+"\n";
231         Middle+="<=X"+QString::number(i)+"<=\n";
232     }
233     Left+="\n";
234     Right+="\n";
235     Middle+="Functions\n";
236     for (auto i = 0u; i < NOutput; ++i) {
237         Left+=QString::number(MinY[i])+"\n";
238         Right+=QString::number(MaxY[i])+"\n";
239         Middle+="<=Y"+QString::number(i)+"<=\n";
240     }
241     ui->LeftBrowser->setText(Left);
242     ui->RightBrowser->setText(Right);
243     ui->MiddleBrowser->setText(Middle);
244     ForwardIsNeed=std::vector<bool>(NInput,true);
245     ForwardIsNeed=std::vector<bool>(NOutput,true);
246 }
247
248 Eigen::VectorXd MainWindow::ForwardDeNormiliseX(Eigen::VectorXd X) {
249     Eigen::VectorXd Result(X);
250     for (auto i = 0u; i < NInput; ++i) {
251         Result(i)=Result(i)*(MaxX[i]-MinX[i])+MinX[i];
252     }
253     return Result;
254 }
255
256 Eigen::VectorXd MainWindow::ForwardDeNormiliseY(Eigen::VectorXd Y) {
257     Eigen::VectorXd Result(Y);
```

```
258     for (auto i = 0u; i < NOutput; ++i) {
259         Result(i)=Result(i)*(MaxY[i]—MinY[i])+MinY[i];
260     }
261     return Result;
262 }
263
264 Eigen::VectorXd MainWindow::ForwardNormiliseX(Eigen::VectorXd X) {
265     Eigen::VectorXd Result(X);
266     for (auto i = 0u; i < NInput; ++i) {
267         if (fabs(MaxX[i]—MinX[i])>1E—20)
268             Result(i)=(Result(i)—MinX[i])/(MaxX[i]—MinX[i]);
269         else
270             Result(i)=1;
271     }
272     return Result;
273 }
274
275 Eigen::VectorXd MainWindow::ForwardNormiliseY(Eigen::VectorXd Y) {
276     Eigen::VectorXd Result(Y);
277     for (auto i = 0u; i < NOutput; ++i) {
278         if (fabs(MaxY[i]—MinY[i])>1E—20)
279             Result(i)=(Result(i)—MinY[i])/(MaxY[i]—MinY[i]);
280         else
281             Result(i)=1;
282     }
283     return Result;
284 }
285
286 MainWindow::~MainWindow()
287 {
288     delete ui;
289 }
290
291 void MainWindow::on_StudyingButton_clicked()
292 {
293     ui—>StudyingButton—>setEnabled(false);
294     ui—>StartButton—>setEnabled(false);
```

```
295     bool XIsEnd = false;
296     std::thread XStudying = std::thread([this,&XIsEnd](){
297         std::vector<bool> ForwardCanGo(NOutput,false);
298         std::vector<std::thread> XThread;
299         for (auto i = 0u; i < NOutput; ++i)
300             XThread.push_back(std::thread([this,i=i,&ForwardCanGo](){
301                 auto Answer = true;
302                 while (Answer)
303                 {
304                     Answer=false;
305                     if (sqrt(ForwardApproximationFunctions[i].Fitness
                        ()/ForwardStudyingData.size())>0.001) {
306                         ForwardApproximationFunctions[i].Studying();
307                         Answer=true;
308                     }
309                 }
310                 ForwardCanGo[i]=true;
311             }));
312         std::thread CheckThread([&,Vec=ForwardCanGo]() {
313             auto ForwardIfCanGo = [&Vec]() {
314                 for (auto i = 0u; i < Vec.size(); ++i)
315                     if (!Vec[i])
316                         return true;
317                 return false;
318             };
319             do {
320                 std::this_thread::sleep_for(std::chrono::milliseconds
                    (25));
321                 Double max=0;
322                 for (auto i = 0u; i < NOutput; ++i) {
323                     auto temp = sqrt(ForwardApproximationFunctions[i
                        ].Fitness()/ForwardStudyingData.size());
324                     if (max<temp)
325                         max=temp;
326                 }
327                 ui->ForwardCurAvErValue->setNum(max);
328             } while (ForwardIfCanGo());
```

```
329         });
330         CheckThread.detach();
331         XThread[0].join();
332         XThread[1].join();
333         XIsEnd=true;
334     });
335     std::thread YStudying = std::thread([this,&XIsEnd](){
336         do {
337             std::this_thread::sleep_for(std::chrono::microseconds(25)
                    );
338         } while(!XIsEnd);
339         auto FirstThread = std::thread([this](){
340             auto Answer = true;
341             while (Answer)
342             {
343                 Answer=false;
344                 for (auto i = 0u; i < NInput/4; ++i)
345                     if (sqrt(BackApproximationFunctions[i].Fitness()/
                           ForwardStudyingData.size())>0.001) {
346                         BackApproximationFunctions[i].Studying();
347                         Answer=true;
348                     }
349             }
350         });
351         auto SecondThread = std::thread([this](){
352             auto Answer = true;
353             while (Answer)
354             {
355                 Answer=false;
356                 for (auto i = NInput/4; i < NInput/2; ++i)
357                     if (sqrt(BackApproximationFunctions[i].Fitness()/
                           ForwardStudyingData.size())>0.001) {
358                         BackApproximationFunctions[i].Studying();
359                         Answer=true;
360                     }
361             }
362         });
```

```
363         auto ThirdThread = std::thread([this](){
364             auto Answer = true;
365             while (Answer)
366             {
367                 Answer=false;
368                 for (auto i = NInput/2; i < 3*NInput/4; ++i)
369                     if (sqrt(BackApproximationFunctions[i].Fitness()/
                           ForwardStudyingData.size())>0.001) {
370                         BackApproximationFunctions[i].Studying();
371                         Answer=true;
372                     }
373             }
374         });
375         auto FourthThread = std::thread([this](){
376             auto Answer = true;
377             while (Answer)
378             {
379                 Answer=false;
380                 for (auto i = 3*NInput/4; i < NInput; ++i)
381                     if (sqrt(BackApproximationFunctions[i].Fitness()/
                           ForwardStudyingData.size())>0.001) {
382                         BackApproximationFunctions[i].Studying();
383                         Answer=true;
384                     }
385             }
386         });
387         std::thread CheckThread([this]() {
388             Double max=0;
389             do {
390                 max=sqrt(SumVecOfNN(BackApproximationFunctions)/
                       ForwardStudyingData.size()/NInput);
391                 std::this_thread::sleep_for(std::chrono::milliseconds
                       (25));
392                 ui->BackCurAvErValue->setNum(max);
393             } while (max>0.001);
394             ui->ForwardCurAvErValue->setNum(max);
395             StatusStream +=("Studying is over.\n");
```

```
396             StatusStream +=(”Neural Networks no need to study.\n”);
397             ui—>StudyingButton—>setEnabled(true);
398             ui—>StartButton—>setEnabled(true);
399         });
400         CheckThread.join();
401         FirstThread.join();
402         SecondThread.join();
403         ThirdThread.join();
404         FourthThread.join();
405     });
406     XStudying.detach();
407     YStudying.detach();
408 }
409
410 Eigen::VectorXd MainWindow::ForwardValue(Eigen::VectorXd X) {
411     Eigen::VectorXd Input(NInput+1);
412     for (auto i = 0u ; i < NInput; ++i)
413         Input[i]=X[i];
414     Input[NInput]=1;
415     Eigen::VectorXd Result(NOutput);
416     for (auto i = 0u; i < NOutput; ++i)
417         Result[i]=ForwardApproximationFunctions[i].Value(Input)[0];
418     return Result;
419 }
420
421 Eigen::VectorXd MainWindow::BackValue(Eigen::VectorXd Y) {
422     Eigen::VectorXd Output(NOutput+1);
423     for (auto i = 0u ; i < NOutput; ++i)
424         Output[i]=Y[i];
425     Output[NInput]=1;
426     Eigen::VectorXd Result(NInput);
427     for (auto i = 0u; i < NInput; ++i) {
428         Result[i]=BackApproximationFunctions[i].Value(Output)[0];
429     }
430     return Result;
431 }
432
```

```
433 void MainWindow::on_StartButton_clicked()
434 {
435     ui->StartButton->setEnabled(false);
436     ui->StudyingButton->setEnabled(false);
437     std::thread ParetoFinding = std::thread ([this](){
438         auto IsIn = [](const Eigen::VectorXd &Low, const Eigen::
               VectorXd &High, const Eigen::VectorXd &X) -> bool {
439             for (auto i = 0u; i < Low.rows(); ++i)
440                 if (Low[i]>X[i] || X[i]>High[i])
441                     return false;
442             return true;
443         };
444         auto HighRand = []() {
445             Double Result = 0.;
446             for (auto i = 1u; i < 15; ++i)
447                 Result+=(rand()%10)/pow(10,i);
448             return Result;
449         };
450         auto IfExistTrue = [](const std::vector<bool> &Vec){
451             for (auto i = 0u; i < Vec.size(); ++i)
452                 if (Vec[i])
453                     return true;
454             return false;
455         };
456         uint EpsilonU = 3;
457         double ProbOfIncluding = 0.005;
458         bool XChanged=false,YChanged=false;
459         std::pair<Eigen::VectorXd,Eigen::VectorXd> XLast = std::pair<
               Eigen::VectorXd,Eigen::VectorXd>(LowX,HighX);
460         std::pair<Eigen::VectorXd,Eigen::VectorXd> YLast = std::pair<
               Eigen::VectorXd,Eigen::VectorXd>(LowY,HighY);
461         do {
462             XChanged=false;
463             YChanged=false;
464             ForwardIsNeed=std::vector<bool>(NInput,true);
465             BackIsNeed=std::vector<bool>(NOutput,true);
466             while (IfExistTrue(ForwardIsNeed)) {
```

```
467                 std::vector<Double> Max(NInput,0);
468                 std::vector<Double> Left(NInput,0),Right(NInput,0);
469                 for (auto i = 0u; i < NInput; ++i) // Forward check
470                     if (ForwardIsNeed[i]) {
471                         Eigen::VectorXd CurrEps = (HighX—LowX)/
                               EpsilonU;
472                         CurrEps[i]/=4;
473                         Eigen::VectorXd Middle=(HighX+LowX)/2;
474                         auto F = [&EpsilonU,
475                                 &HighRand,
476                                 &IsIn,
477                                 &ProbOfIncluding,
478                                 &Middle]
479                                 (const Eigen::VectorXd &LowY,
480                                 const Eigen::VectorXd &HighY,
481                                 const Eigen::VectorXd &Low,
482                                 const Eigen::VectorXd &High,
483                                 const Eigen::VectorXd &CurrEps,
484                                 const std::vector<NeuralNetwork> &
                                       ApproxFunc) —> uint {
485                             uint BettaSum = 0;
486                             Double Delta = 0;
487                             {
488                                 Eigen::VectorXd CurrVec(Low);
489                                 do {
490                                     for (auto s = 0u; s < Low.rows();
                                           ++s)
491                                         if (CurrVec[s]<High[s]—0.5*
                                               CurrEps[s]) {
492                                             CurrVec[s]+=CurrEps[s];
493                                             break;
494                                         }
495                                         else
496                                             CurrVec[s]=Low[s];
497                                     Delta+=1/(Middle—CurrVec).norm();
498                                 } while ((CurrVec—High).norm()>geps);
499                             }
```

```
500                         Eigen::VectorXd CurrVec(Low);
501                         uint Sum = 1;
502                         for (auto g = 0u; g < CurrEps.rows(); ++g
                               )
503                             Sum*=EpsilonU;
504                         Eigen::VectorXd YMiddle = (HighY+LowY)/2;
505                         do {
506                             for (auto s = 0u; s < Low.rows(); ++s
                                   )
507                                 if (CurrVec[s]<High[s]-0.5*
                                       CurrEps[s]) {
508                                     CurrVec[s]+=CurrEps[s];
509                                     break;
510                                 }
511                                 else
512                                     CurrVec[s]=Low[s];
513                             Eigen::VectorXd Y(LowY);
514                             if (HighRand()<Sum*ProbOfIncluding/(
                                   Delta*(Middle-CurrVec).norm())) {
515                                 Eigen::VectorXd Input(CurrVec.
                                       rows()+1);
516                                 for (auto p = 0u; p < CurrVec.
                                       rows(); ++p)
517                                     Input[p]=CurrVec[p];
518                                 Input[CurrVec.rows()]=1;
519                                 for (auto z = 0u; z < LowY.size()
                                       ; ++z)
520                                     Y[z]=ApproxFunc[z].Value(
                                           Input)[0];
521                                 if(!IsIn(LowY,HighY,Y))
522                                     ++BettaSum;
523                             }
524                         } while ((CurrVec-High).norm()>geps);
525                         return BettaSum;
526                     };
527                     //left side
528                     {
```

```
529                         Eigen::VectorXd LeftLowX=LowX;
530                         Eigen::VectorXd LeftHighX=HighX;
531                         LeftHighX[i]=LeftLowX[i]*0.75+0.25*HighX[
                               i];
532                         Eigen::VectorXd CurrVec(LeftLowX);
533                         Left[i] = F(LowY,HighY,LeftLowX,LeftHighX
                               ,CurrEps,ForwardApproximationFunctions
                               );
534                     }
535                     //right side
536                     {
537                         Eigen::VectorXd RightLowX=LowX;
538                         Eigen::VectorXd RightHighX=HighX;
539                         RightLowX[i]=RightLowX[i]*0.25+0.75*
                               RightHighX[i];
540                         Eigen::VectorXd CurrVec(RightLowX);
541                         Right[i] = F(LowY,HighY,RightLowX,
                               RightHighX,CurrEps,
                               ForwardApproximationFunctions);
542                     }
543                     if (Right[i]>0 || Left[i]>0) {
544                         if (Left[i]<Right[i]) {
545                             HighX[i]-=0.01*(HighX[i]-LowX[i]);
546                             StatusStream +=("X"+QString::number(i
                                   )+" was redused from right (" +
                                   QString::number(Left[i]) + "<" +
                                   QString::number(Right[i]) + ")\n")
                                   ;
547                         }
548                         else {
549                             LowX[i]+=0.01*(HighX[i]-LowX[i]);
550                             StatusStream +=("X"+QString::number(i
                                   )+" was redused from left (" +
                                   QString::number(Left[i]) + ">" +
                                   QString::number(Right[i]) + ")\n")
                                   ;
551                         }
```

```
552                         QString Left="",Middle="",Right="";
553                         Left+="\n";
554                         Right+="\n";
555                         Middle+="Parametrs\n";
556                         for (auto i = 0u; i < NInput; ++i) {
557                             Left+=QString::number(LowX[i]*(MaxX[i
                                ]-MinX[i])+MinX[i])+"\n";
558                             Right+=QString::number(HighX[i]*(MaxX
                                [i]-MinX[i])+MinX[i])+"\n";
559                             Middle+="<=X"+QString::number(i)+"<=\
                                n";
560                         }
561                         Left+="\n";
562                         Right+="\n";
563                         Middle+="Functions\n";
564                         for (auto i = 0u; i < NOutput; ++i) {
565                             Left+=QString::number(LowY[i]*(MaxY[i
                                ]-MinY[i])+MinY[i])+"\n";
566                             Right+=QString::number(HighY[i]*(MaxY
                                [i]-MinY[i])+MinY[i])+"\n";
567                             Middle+="<=Y"+QString::number(i)+"<=\
                                n";
568                         }
569                         ui->LeftBrowser->setText(Left);
570                         ui->RightBrowser->setText(Right);
571                         ui->MiddleBrowser->setText(Middle);
572                         XChanged=true;
573                     }
574                     else {
575                         StatusStream += "X"+QString::number(i)+"
                                no need to change on this iteration\n
                                ";
576                         ForwardIsNeed[i]=false;
577                     }
578                 }
579             }
580             ProbOfIncluding = 0.05;
```

```
581             while (IfExistTrue(BackIsNeed)) {
582                 std::vector<Double> Max(NOutput,0);
583                 std::vector<Double> Left(NOutput,0),Right(NOutput,0);
584                 for (auto i = 0u; i < NOutput; ++i) // Back check
585                     if (BackIsNeed[i]) {
586                         Eigen::VectorXd CurrEps = 0.005*(HighY-LowY)/
                               EpsilonU;
587                         CurrEps[i]/=4;
588                         Eigen::VectorXd Middle=(HighY+LowY)/2;
589                         auto F = [&EpsilonU,
590                                 &HighRand,
591                                 &IsIn,
592                                 &ProbOfIncluding,
593                                 &Middle,
594                                 &Max,
595                                 I=i]
596                                 (const Eigen::VectorXd &LowY,
597                                 const Eigen::VectorXd &HighY,
598                                 const Eigen::VectorXd &Low,
599                                 const Eigen::VectorXd &High,
600                                 const Eigen::VectorXd &CurrEps,
601                                 const std::vector<NeuralNetwork> &
                                       ApproxFunc) -> uint {
602                             uint BettaSum = 0;
603                             Double Delta = 0;
604                             {
605                                 Eigen::VectorXd CurrVec(Low);
606                                 do {
607                                     for (auto s = 0u; s < Low.rows();
                                           ++s)
608                                         if (CurrVec[s]<High[s]-0.5*
                                               CurrEps[s]) {
609                                             CurrVec[s]+=CurrEps[s];
610                                             break;
611                                         }
612                                         else
613                                             CurrVec[s]=Low[s];
```

```
614                         Delta+=1/(Middle-CurrVec).norm();
615                     } while ((CurrVec-High).norm()>geps);
616                 }
617                 Eigen::VectorXd CurrVec(Low);
618                 uint Sum = 1;
619                 for (auto g = 0u; g < CurrEps.rows(); ++g
                        )
620                     Sum*=EpsilonU;
621                 Eigen::VectorXd YMiddle = (HighY+LowY)/2;
622                 do {
623                     for (auto s = 0u; s < Low.rows(); ++s
                            )
624                         if (CurrVec[s]<High[s]-0.5*
                                CurrEps[s]) {
625                             CurrVec[s]+=CurrEps[s];
626                             break;
627                         }
628                         else
629                             CurrVec[s]=Low[s];
630                     Eigen::VectorXd Y(LowY);
631                     if (HighRand()<Sum*ProbOfIncluding/(
                            Delta*(Middle-CurrVec).norm())) {
632                         Eigen::VectorXd Input(CurrVec.
                                rows()+1);
633                         for (auto p = 0u; p < CurrVec.
                                rows(); ++p)
634                             Input[p]=CurrVec[p];
635                         Input[CurrVec.rows()]=1;
636                         for (auto z = 0u; z < LowY.size()
                                ; ++z)
637                             Y[z]=ApproxFunc[z].Value(
                                    Input)[0];
638                         if(!IsIn(LowY,HighY,Y))
639                             ++BettaSum;
640                         Y-=YMiddle;
641                         for (auto z = 0u; z < LowY.size()
                                ; ++z)
```

```
642                                 if (Max[I]<fabs(Y[z]))
643                                     Max[I]=fabs(Y[z]);
644                             }
645                         } while ((CurrVec—High).norm()>geps);
646                         return BettaSum;
647                     };
648                     //left side
649                     {
650                         Eigen::VectorXd LeftLowY=LowY;
651                         Eigen::VectorXd LeftHighY=HighY;
652                         LeftHighY[i]=LeftLowY[i]*0.75+0.25*HighY[
                               i];
653                         Eigen::VectorXd CurrVec(LeftLowY);
654                         Left[i] = F(LowX,HighX,LeftLowY,LeftHighY
                               ,CurrEps,BackApproximationFunctions);
655                     }
656                     //right side
657                     {
658                         Eigen::VectorXd RightLowY=LowY;
659                         Eigen::VectorXd RightHighY=HighY;
660                         RightLowY[i]=RightLowY[i]*0.25+0.75*
                               RightHighY[i];
661                         Eigen::VectorXd CurrVec(RightLowY);
662                         Right[i] = F(LowX,HighX,RightLowY,
                               RightHighY,CurrEps,
                               BackApproximationFunctions);
663                     }
664                     if (Right[i]>0 || Left[i]>0) {
665                         if (Left[i]<Right[i]) {
666                             HighY[i]—=0.01*(HighY[i]—LowY[i]);
667                             StatusStream +=(”Y”+QString::number(i
                                   )+” was redused from right (” +
                                   QString::number(Left[i]) + ”<” +
                                   QString::number(Right[i]) + ”)\n”)
                                   ;
668                         }
669                         else {
```

```
670                         LowY[i]+=0.01*(HighY[i]—LowY[i]);
671                         StatusStream +=(”Y”+QString::number(i
                               )+” was redused from left (” +
                               QString::number(Left[i]) + ”>” +
                               QString::number(Right[i]) + ”)\n”)
                               ;
672                     }
673                     QString Left=””,Middle=””,Right=””;
674                     Left+=”\n”;
675                     Right+=”\n”;
676                     Middle+=”Parametrs\n”;
677                     for (auto i = 0u; i < NInput; ++i) {
678                         Left+=QString::number(LowX[i]*(MaxX[i
                               ]—MinX[i])+MinX[i])+”\n”;
679                         Right+=QString::number(HighX[i]*(MaxX
                               [i]—MinX[i])+MinX[i])+”\n”;
680                         Middle+=”<=X”+QString::number(i)+”<=\
                               n”;
681                     }
682                     Left+=”\n”;
683                     Right+=”\n”;
684                     Middle+=”Functions\n”;
685                     for (auto i = 0u; i < NOutput; ++i) {
686                         Left+=QString::number(LowY[i]*(MaxY[i
                               ]—MinY[i])+MinY[i])+”\n”;
687                         Right+=QString::number(HighY[i]*(MaxY
                               [i]—MinY[i])+MinY[i])+”\n”;
688                         Middle+=”<=Y”+QString::number(i)+”<=\
                               n”;
689                     }
690                     ui—>LeftBrowser—>setText(Left);
691                     ui—>RightBrowser—>setText(Right);
692                     ui—>MiddleBrowser—>setText(Middle);
693                     YChanged=true;
694                 }
695                 else {
```

```
696                             StatusStream += ”Y”+QString::number(i)+”
                                   no need to change on this iteration\n
                                   ”;
697                             BackIsNeed[i]=false;
698                         }
699                     }
700             }
701             if ((XLast.second—XLast.first—HighX+LowX).norm()<5E—2 &&
702                     (YLast.second—YLast.first—(HighY—LowY)).norm()<5E
                           —2)
703                 break;
704             else {
705                 XLast = std::pair<Eigen::VectorXd,Eigen::VectorXd>(
                       LowX,HighX);
706                 YLast = std::pair<Eigen::VectorXd,Eigen::VectorXd>(
                       LowY,HighY);
707             }
708         } while (XChanged||YChanged);
709         ui—>StartButton—>setEnabled(true);
710         ui—>StudyingButton—>setEnabled(true);
711     });
712     ParetoFinding.detach();
713 }
714
715 void MainWindow::on_ShowGraph_clicked()
716 {
717     for (auto Q = 0u; Q < NOutput; ++Q) {
718         QVector<double> XApp,YApp,Y;
719         auto plt = new QCustomPlot;
720         double kek = —1;
721         for (auto q = 0u; q < ForwardStudyingData[0].size(); ++q) {
722             XApp.push_back(++kek);
723             YApp.push_back(ForwardApproximationFunctions[Q].Value(
                       ForwardStudyingData[Q][q].first)[0]);
724             Y.push_back(ForwardStudyingData[Q][q].second[0]);
725         }
726         plt—>addGraph();
```

```
727         plt->addGraph();
728         plt->graph(0)->setData(XApp,Y);
729         plt->graph(1)->setData(XApp,YApp);
730         plt->graph(1)->setPen(QPen(Qt::red));
731         plt->resize(600, 600);
732         plt->rescaleAxes(1);
733         plt->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom);
734         plt->show();
735     }
736     for (auto Q = 0u; Q < NInput; ++Q) {
737         QVector<double> XApp,YApp,Y;
738         auto plt = new QCustomPlot;
739         double kek = -1;
740         for (auto q = 0u; q < BackStudyingData[0].size(); ++q) {
741             XApp.push_back(++kek);
742             YApp.push_back(BackApproximationFunctions[Q].Value(
                    BackStudyingData[Q][q].first)[0]);
743             Y.push_back(BackStudyingData[Q][q].second[0]);
744         }
745         plt->addGraph();
746         plt->addGraph();
747         plt->graph(0)->setData(XApp,Y);
748         plt->graph(1)->setData(XApp,YApp);
749         plt->graph(1)->setPen(QPen(Qt::red));
750         plt->resize(600, 600);
751         plt->rescaleAxes(1);
752         plt->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom);
753         plt->show();
754     }
755 }
```

```
1 #ifndef NEURALNETWORKPERCEPTRON_H
2 #define NEURALNETWORKPERCEPTRON_H
3
4 #include <iostream>
5 #include <eigen3/Eigen/Dense>
6 #include <vector>
7 #include <cstdlib>
```

```
 8 #include <ctime>
 9 #include <QVector>
10 #include <string>
11 #include <fstream>
12 #include <iomanip>
13
14 #define geps 1E—6
15 #define geps2 1E—5
16
17 using Double = double;
18
19 Double norm_dist();
20
21
22 class NeuralNetwork {
23 private:
24     std::vector<Eigen::MatrixXd> LayerConvertMatrix;
25     std::function<Double(Double)> SigmaFunction;
26     std::vector<std::pair<Eigen::VectorXd,Eigen::VectorXd>>
          TeacherExamples;
27     std::string ValueFileName;
28 public:
29     NeuralNetwork(){}
30     Double Fitness();
31     Double Fitness(const std::vector<Eigen::MatrixXd>& Matrixs);
32     NeuralNetwork(std::vector<Eigen::MatrixXd> LayerConvertMatrix_,
33                   std::function<Double(Double)> SigmaFunction_,
34                   std::vector<std::pair<Eigen::VectorXd,Eigen::
                        VectorXd>> TeacherExamples_,
35                   std::string ValueFileName_);
36     Eigen::VectorXd Value(const Eigen::VectorXd &X) const;
37     void Studying();
38     ~NeuralNetwork(){}
39 };
40
41 Double SumVecOfNN(std::vector<NeuralNetwork> Vec);
42
```

```
43 #endif // NEURALNETWORKPERCEPTRON_H
```

```
 1 #include ”NeuralNetworkPerceptron.h”
 2
 3 Double norm_dist() {
 4     Double res = 0;
 5     for (auto i = 0u; i < 12; ++i)
 6         res+=rand()%10001/10000.−0.5;
 7     return res;
 8 }
 9
10 NeuralNetwork::NeuralNetwork(std::vector<Eigen::MatrixXd>
      LayerConvertMatrix_,
11                              std::function<Double(Double)>
                                   SigmaFunction_,
12                              std::vector<std::pair<Eigen::VectorXd,
                                   Eigen::VectorXd>> TeacherExamples_,
13                              std::string ValueFileName_) {
14     LayerConvertMatrix=LayerConvertMatrix_;
15     SigmaFunction=SigmaFunction_;
16     TeacherExamples=TeacherExamples_;
17     ValueFileName=ValueFileName_;
18 }
19
20 Eigen::VectorXd NeuralNetwork::Value(const Eigen::VectorXd &X) const
      {
21     auto Result=X;
22     for (auto i = 0u; i < LayerConvertMatrix.size(); ++i) {
23         Result=LayerConvertMatrix[i]*Result;
24         if (i!=LayerConvertMatrix.size()−1)
25             for (auto j = 0u; j < Result.size(); ++j)
26                 Result[j]=SigmaFunction(Result[j]);
27     }
28     return Result;
29 }
30
31 void NeuralNetwork::Studying() {
32     Double Golden = (1+sqrt(5))/2;
```

```
33      auto X = LayerConvertMatrix,X0(X);
34      uint NTacts = 0u;
35      for (auto i = 0u; i < LayerConvertMatrix.size(); ++i)
36          NTacts+=LayerConvertMatrix[i].rows()*LayerConvertMatrix[i].
               cols();
37      std::vector<Double> LayerProb(LayerConvertMatrix.size());
38      Double Sum = 0;
39      for (auto i = 0u; i < LayerConvertMatrix.size(); ++i) {
40          LayerProb[i]=LayerConvertMatrix[i].rows()*LayerConvertMatrix[
               i].cols();
41          Sum+=LayerProb[i];
42      }
43      for (auto i = 0u; i < LayerConvertMatrix.size(); ++i) {
44          LayerProb[i]/=Sum;
45      }
46      auto RightMatrix(X),LeftMatrix(X);
47      for (auto H = 0u; H < NTacts; ++H) {
48          uint i = 0;
49          auto CurrProb=rand()%10000/10000.;
50          for (auto z = 0u; z < LayerConvertMatrix.size(); ++z) {
51              if (CurrProb-LayerProb[z]>0)
52                  CurrProb-=LayerProb[z];
53              else
54                  i=z;
55          }
56          uint j = uint(rand())%LayerConvertMatrix[i].rows();
57          uint k = uint(rand())%LayerConvertMatrix[i].cols();
58          double Differential;
59          RightMatrix[i](j,k)+=geps;
60          LeftMatrix[i](j,k)-=geps;
61          Differential=(Fitness(RightMatrix)-Fitness(LeftMatrix))/(2*
               geps);
62          auto A = 0.,B = 1.;
63          auto Curr=Fitness(X0);
64          do {
65              B/=2;
66              X[i](j,k)=X0[i](j,k)-B*Differential;
```

```
 67          }
 68          while (Fitness(X)>Curr);
 69          X0[i](j,k)=X[i](j,k);
 70          RightMatrix[i](j,k)=X0[i](j,k);
 71          LeftMatrix[i](j,k)=X0[i](j,k);
 72      }
 73      std::ofstream fout(ValueFileName+".txt");
 74      fout << LayerConvertMatrix.size() << " ";
 75      fout << LayerConvertMatrix[0].cols() << " ";
 76      for (auto i = 0u; i < LayerConvertMatrix.size(); ++i)
 77          fout << LayerConvertMatrix[i].rows() << ' ';
 78      fout << '\n';
 79      for (auto i = 0u; i < X.size(); ++i) {
 80          for (auto j = 0u; j < X[i].rows(); ++j) {
 81              for (auto k = 0u; k < X[i].cols(); ++k)
 82                  fout << std::fixed << std::setprecision(15) << X0[i](
                        j,k) << ' ';
 83              fout << '\n';
 84          }
 85      }
 86      fout.close();
 87      LayerConvertMatrix=X0;
 88  }
 89
 90  //void NeuralNetwork::Studying() {
 91  //    Double Golden = (1+sqrt(5))/2;
 92  //    auto X = LayerConvertMatrix,X0(X);
 93  //    X0=X;
 94  //    auto DifferentialLayerConvertMatrix(X);
 95  //    for (auto i = 0u; i < LayerConvertMatrix.size(); ++i)
 96  //        for (auto j = 0u; j < LayerConvertMatrix[i].rows(); ++j)
 97  //            for (auto k = 0u; k < LayerConvertMatrix[i].cols(); ++k
       ) {
 98  //                auto RightMatrix(X),LeftMatrix(X);
 99  //                RightMatrix[i](j,k)+=geps;
100  //                LeftMatrix[i](j,k)—=geps;
```

```
101 //                DifferentialLayerConvertMatrix[i](j,k)=(Fitness(
    RightMatrix)—Fitness(LeftMatrix))/(2*geps)+norm_dist()/48;
102 //            }
103 //    auto A(LayerConvertMatrix),B(LayerConvertMatrix);
104 //    for (auto i = 0u; i < LayerConvertMatrix.size(); ++i)
105 //        for (auto j = 0u; j < LayerConvertMatrix[i].rows(); ++j)
106 //            for (auto k = 0u; k < LayerConvertMatrix[i].cols(); ++k
    ) {
107 //                A[i](j,k)=0;
108 //                B[i](j,k)=1;
109 //            }
110 //    auto X_(X);
111 //    for (auto i = 0u; i < LayerConvertMatrix.size(); ++i)
112 //        for (auto j = 0u; j < LayerConvertMatrix[i].rows(); ++j)
113 //            for (auto k = 0u; k < LayerConvertMatrix[i].cols(); ++k
    ) {
114 //                do {
115 //                    auto X1=B[i](j,k)—(B[i](j,k)—A[i](j,k))/Golden;
116 //                    auto X2=A[i](j,k)+(B[i](j,k)—A[i](j,k))/Golden;
117 //                    Double Y1,Y2;
118 //                    X[i](j,k)=X0[i](j,k)—X1*
    DifferentialLayerConvertMatrix[i](j,k);
119 //                    Y1=Fitness(X);
120 //                    X[i](j,k)=X0[i](j,k)—X2*
    DifferentialLayerConvertMatrix[i](j,k);
121 //                    Y2=Fitness(X);
122 //                    if (Y1>Y2)
123 //                        A[i](j,k)=X1;
124 //                    else
125 //                        B[i](j,k)=X2;
126 //                }
127 //                while (B[i](j,k)—A[i](j,k)>geps2);
128 //                auto alpha=(A[i](j,k)+B[i](j,k))/2;
129 //                X_[i](j,k)=X0[i](j,k)—alpha*
    DifferentialLayerConvertMatrix[i](j,k);
130 //            }
131 //    std::ofstream fout(ValueFileName+".txt");
```

```
132 //    fout << LayerConvertMatrix.size() << " ";
133 //    fout << LayerConvertMatrix[0].cols() << " ";
134 //    for (auto i = 0u; i < LayerConvertMatrix.size(); ++i)
135 //        fout << LayerConvertMatrix[i].rows() << ' ';
136 //    fout << '\n';
137 //    for (auto i = 0u; i < X.size(); ++i) {
138 //        for (auto j = 0u; j < X[i].rows(); ++j) {
139 //            for (auto k = 0u; k < X[i].cols(); ++k)
140 //                fout << std::fixed << std::setprecision(15) << X[i
       ](j,k) << ' ';
141 //            fout << '\n';
142 //        }
143 //    }
144 //    fout.close();
145 //    LayerConvertMatrix=X_;
146 //}
147
148 Double NeuralNetwork::Fitness (const std::vector<Eigen::MatrixXd>&
       Matrixs) {
149     Double Result = 0;
150     for (auto i = 0u; i < TeacherExamples.size(); ++i) {
151         Eigen::VectorXd VecResult = TeacherExamples[i].first;
152         for (auto j = 0u; j < Matrixs.size(); ++j) {
153             VecResult=Matrixs[j]*VecResult;
154             if (j!=Matrixs.size()-1)
155                 for (auto k = 0u; k < VecResult.size(); ++k)
156                     VecResult[k]=SigmaFunction(VecResult[k]);
157         }
158         Result+=(VecResult-TeacherExamples[i].second).squaredNorm();
159     }
160     return Result;
161 };
162
163 Double NeuralNetwork::Fitness() {
164     Double Result = 0;
165     for (auto i = 0u; i < TeacherExamples.size(); ++i) {
166         Eigen::VectorXd VecResult = TeacherExamples[i].first;
```

```
167         for (auto j = 0u; j < LayerConvertMatrix.size(); ++j) {
168             VecResult=LayerConvertMatrix[j]*VecResult;
169             if (j!=LayerConvertMatrix.size()-1)
170                 for (auto k = 0u; k < VecResult.size(); ++k)
171                     VecResult[k]=SigmaFunction(VecResult[k]);
172         }
173         Result+=(VecResult-TeacherExamples[i].second).squaredNorm();
174     }
175     return Result;
176 };
177
178 Double SumVecOfNN(std::vector<NeuralNetwork> Vec) {
179     Double res = 0;
180     for (auto &it : Vec)
181         res+=it.Fitness();
182     return res;
183 }
```

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ui version="4.0">
3   <class>MainWindow</class>
4   <widget class="QMainWindow" name="MainWindow">
5    <property name="geometry">
6     <rect>
7      <x>0</x>
8      <y>0</y>
9      <width>976</width>
10     <height>673</height>
11    </rect>
12   </property>
13   <property name="windowTitle">
14    <string>MainWindow</string>
15   </property>
16   <widget class="QWidget" name="centralwidget">
17    <widget class="QWidget" name="gridLayoutWidget">
18     <property name="geometry">
19      <rect>
20       <x>10</x>
```

```
21        <y>10</y>
22        <width>951</width>
23        <height>621</height>
24       </rect>
25      </property>
26      <layout class="QGridLayout" name="gridLayout">
27       <item row="2" column="1">
28        <layout class="QHBoxLayout" name="horizontalLayout_8">
29         <item>
30          <spacer name="verticalSpacer">
31           <property name="orientation">
32            <enum>Qt::Vertical</enum>
33           </property>
34           <property name="sizeHint" stdset="0">
35            <size>
36             <width>20</width>
37             <height>300</height>
38            </size>
39           </property>
40          </spacer>
41         </item>
42         <item>
43          <spacer name="LeftSpacer">
44           <property name="orientation">
45            <enum>Qt::Horizontal</enum>
46           </property>
47           <property name="sizeHint" stdset="0">
48            <size>
49             <width>400</width>
50             <height>20</height>
51            </size>
52           </property>
53          </spacer>
54         </item>
55         <item>
56          <widget class="QLabel" name="LeftBrowser">
57           <property name="text">
```

```
58            <string>TextLabel</string>
59           </property>
60          </widget>
61         </item>
62         <item>
63          <widget class="QLabel" name="MiddleBrowser">
64           <property name="text">
65            <string>TextLabel</string>
66           </property>
67          </widget>
68         </item>
69         <item>
70          <widget class="QLabel" name="RightBrowser">
71           <property name="text">
72            <string>TextLabel</string>
73           </property>
74          </widget>
75         </item>
76         <item>
77          <spacer name="RightSpacer">
78           <property name="orientation">
79            <enum>Qt::Horizontal</enum>
80           </property>
81           <property name="sizeHint" stdset="0">
82            <size>
83             <width>400</width>
84             <height>20</height>
85            </size>
86           </property>
87          </spacer>
88         </item>
89         <item>
90          <spacer name="verticalSpacer_2">
91           <property name="orientation">
92            <enum>Qt::Vertical</enum>
93           </property>
94           <property name="sizeHint" stdset="0">
```

```
 95           <size>
 96            <width>20</width>
 97            <height>300</height>
 98           </size>
 99          </property>
100         </spacer>
101        </item>
102       </layout>
103      </item>
104      <item row="3" column="1">
105       <layout class="QHBoxLayout" name="horizontalLayout_19">
106        <item>
107         <spacer name="horizontalSpacer_3">
108          <property name="orientation">
109           <enum>Qt::Horizontal</enum>
110          </property>
111          <property name="sizeHint" stdset="0">
112           <size>
113            <width>40</width>
114            <height>20</height>
115           </size>
116          </property>
117         </spacer>
118        </item>
119        <item>
120         <widget class="QLabel" name="label">
121          <property name="text">
122           <string>Status Browser</string>
123          </property>
124         </widget>
125        </item>
126        <item>
127         <spacer name="horizontalSpacer_4">
128          <property name="orientation">
129           <enum>Qt::Horizontal</enum>
130          </property>
131          <property name="sizeHint" stdset="0">
```

```
132           <size>
133            <width>40</width>
134            <height>20</height>
135           </size>
136          </property>
137         </spacer>
138        </item>
139       </layout>
140      </item>
141      <item row="1" column="1">
142       <layout class="QVBoxLayout" name="verticalLayout_2">
143        <item>
144         <layout class="QHBoxLayout" name="horizontalLayout_6">
145          <item>
146           <widget class="QPushButton" name="StudyingButton">
147            <property name="text">
148             <string>Study Neural Network</string>
149            </property>
150           </widget>
151          </item>
152          <item>
153           <spacer name="UpSpacer">
154            <property name="orientation">
155             <enum>Qt::Horizontal</enum>
156            </property>
157            <property name="sizeHint" stdset="0">
158             <size>
159              <width>40</width>
160              <height>20</height>
161             </size>
162            </property>
163           </spacer>
164          </item>
165          <item>
166           <layout class="QVBoxLayout" name="verticalLayout">
167            <item>
168             <layout class="QHBoxLayout" name="horizontalLayout">
```

```
169              <item>
170               <widget class=”QLabel” name=”ForwardCurAvEr”>
171                <property name=”text”>
172                 <string>Curren Forward Average Error</string>
173                </property>
174               </widget>
175              </item>
176              <item>
177               <widget class=”QLabel” name=”ForwardCurAvErValue”>
178                <property name=”text”>
179                 <string>1</string>
180                </property>
181               </widget>
182              </item>
183             </layout>
184            </item>
185            <item>
186             <layout class=”QHBoxLayout” name=”horizontalLayout_3”>
187              <item>
188               <widget class=”QLabel” name=”BackCurAvEr”>
189                <property name=”text”>
190                 <string>Curren back Average Error</string>
191                </property>
192               </widget>
193              </item>
194              <item>
195               <widget class=”QLabel” name=”BackCurAvErValue”>
196                <property name=”text”>
197                 <string>1</string>
198                </property>
199               </widget>
200              </item>
201             </layout>
202            </item>
203           </layout>
204          </item>
205         </layout>
```

```
206         </item>
207         <item>
208          <layout class="QHBoxLayout" name="horizontalLayout_14">
209           <item>
210            <spacer name="horizontalSpacer_2">
211             <property name="orientation">
212              <enum>Qt::Horizontal</enum>
213             </property>
214             <property name="sizeHint" stdset="0">
215              <size>
216               <width>40</width>
217               <height>20</height>
218              </size>
219             </property>
220            </spacer>
221           </item>
222           <item>
223            <widget class="QPushButton" name="StartButton">
224             <property name="text">
225              <string>Find Pareto Set</string>
226             </property>
227            </widget>
228           </item>
229           <item>
230            <spacer name="horizontalSpacer_5">
231             <property name="orientation">
232              <enum>Qt::Horizontal</enum>
233             </property>
234             <property name="sizeHint" stdset="0">
235              <size>
236               <width>40</width>
237               <height>20</height>
238              </size>
239             </property>
240            </spacer>
241           </item>
242           <item>
```

```
243           <widget class="QPushButton" name="ShowGraph">
244            <property name="text">
245             <string>Show Graph</string>
246            </property>
247           </widget>
248          </item>
249          <item>
250           <spacer name="horizontalSpacer">
251            <property name="orientation">
252             <enum>Qt::Horizontal</enum>
253            </property>
254            <property name="sizeHint" stdset="0">
255             <size>
256              <width>40</width>
257              <height>20</height>
258             </size>
259            </property>
260           </spacer>
261          </item>
262         </layout>
263        </item>
264       </layout>
265      </item>
266      <item row="5" column="1">
267       <widget class="QPlainTextEdit" name="StatusBrowser"/>
268      </item>
269     </layout>
270    </widget>
271   </widget>
272   <widget class="QMenuBar" name="menubar">
273    <property name="geometry">
274     <rect>
275      <x>0</x>
276      <y>0</y>
277      <width>976</width>
278      <height>22</height>
279     </rect>
```

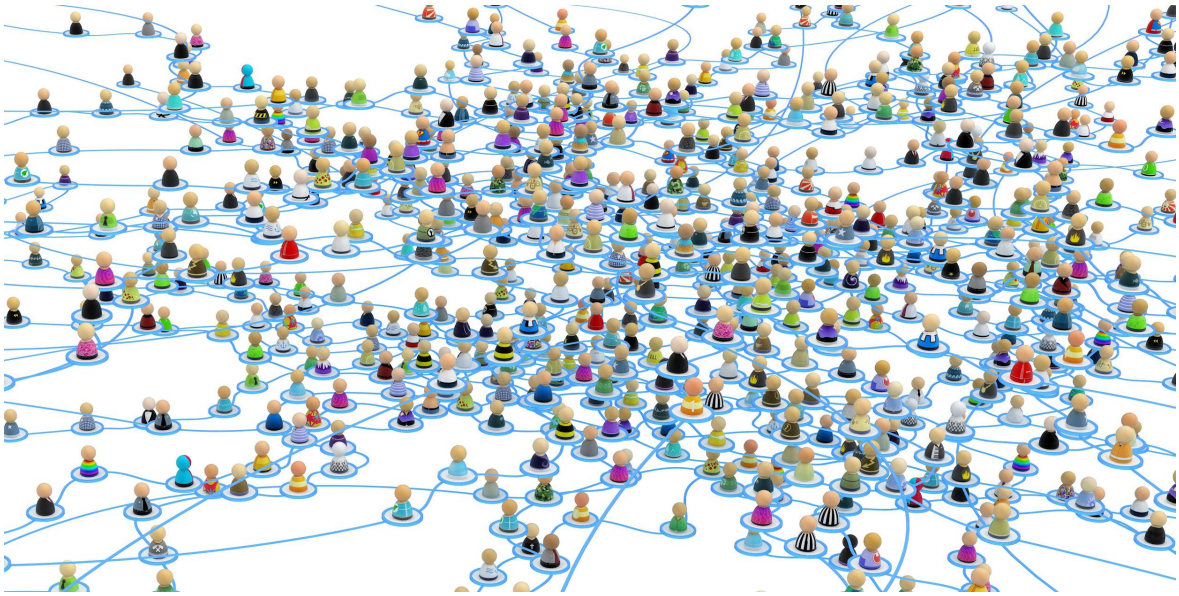
```
280     </property>
281    </widget>
282    <widget class=”QStatusBar” name=”statusbar”/>
283   </widget>
284   <resources/>
285   <connections/>
286  </ui>
```

ДОДАТОК Б ПРЕЗЕНТАЦІЯ

Узгодження множини значень складної проектованої системи з областю визначення внутрішніх параметрів

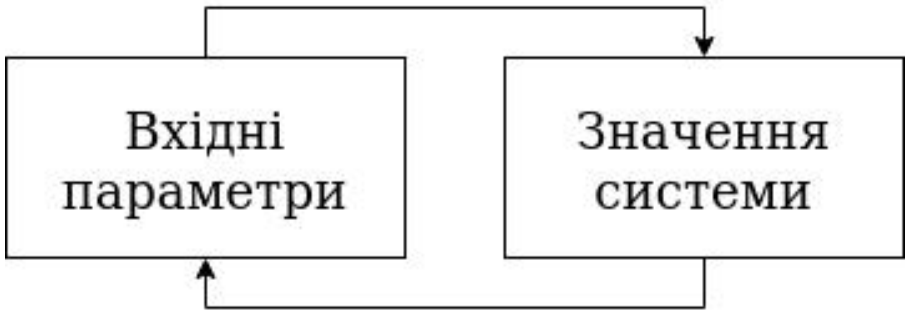
презентація студента групи КА-64
Чикивді Олександра

Складна система, її роль в системному аналізі

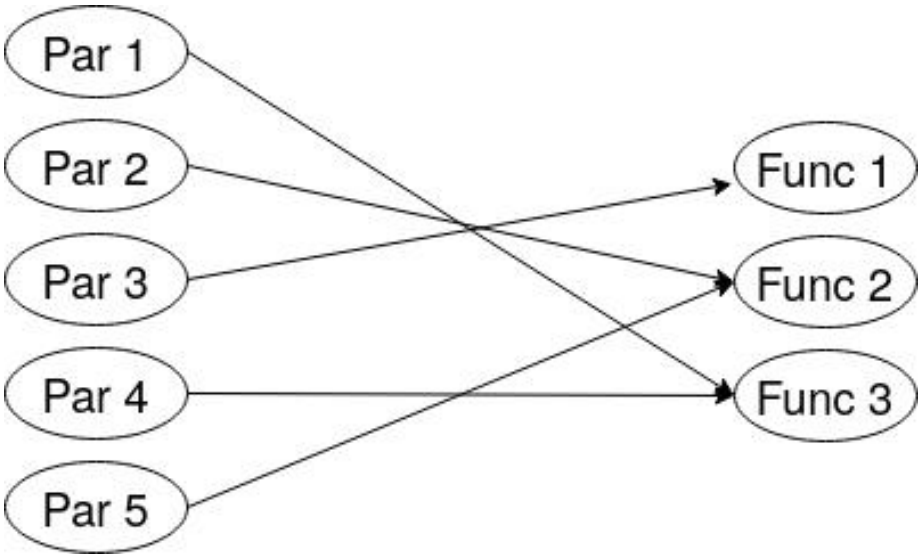


Вхідні параметри та значення системи

Постановка задачі



Постановка задачі



Вхідні параметри та значення системи

Нехай об'єкт функціонує на певному інтервалі $[a,b] \subset R$ та на цьому інтервалі описується вектором стану $Y_t, t \in [a,b]$. Нехай $\{t_i\}_{i=0}^{N_t}$ розбиття інтервалу $[a,b]$ на $N_t + 1$ кроків з кроком $\varepsilon = \frac{b-a}{N_t}$, тобто $\forall i = 0, \ldots, N_t : t_i = a + \varepsilon$. Нехай на цьому розбитті задано множини значень вектора стану $\{Y_i\}_{i=0}^{N_t} = \left\{Y_i^0, \ldots, Y_i^{N_Y}\right\}_{i=0}^{N_t}$, вектора керуючих параметрів $\{U_i\}_{i=0}^{N_t} = \left\{U_i^0, \ldots, U_i^{N_U}\right\}_{i=0}^{N_t}$ та вектора некеруючих параметрів $\{X_i\}_{i=0}^{N_t} = \left\{X_i^0, \ldots, X_i^{N_X}\right\}_{i=0}^{N_t}$. Таку трійку множин та часове розбиття називатимемо системою. Будемо використовувати наступне позначення:$S(Y_{N_y}, U_{N_U}, X_{N_X}, [a,b]_{N_t})$.

Множина Парето

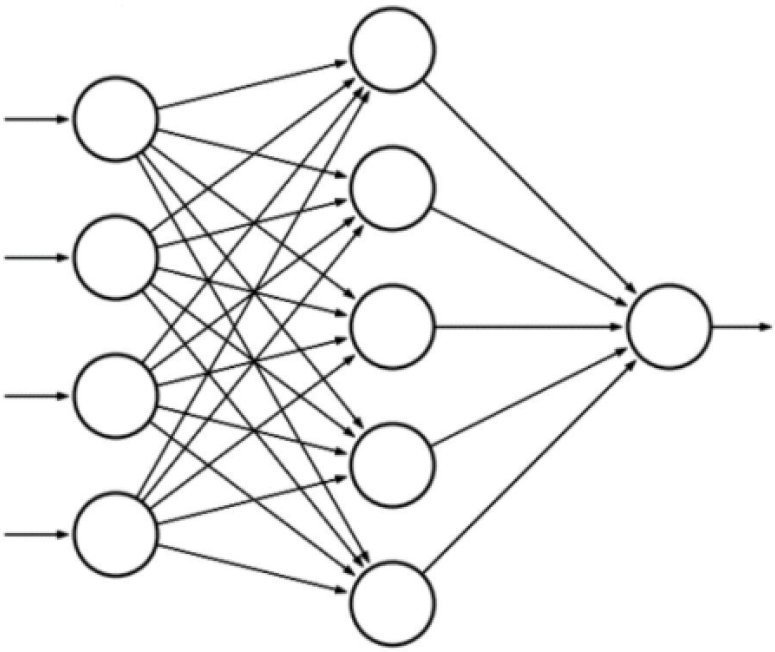
Множиною Парето будемо називати таку трійку $\{P_Y^*, P_U^*, P_X^*\}$:

$$P_Y^* = \{[a_0^*, b_0^*]_Y, \ldots, [a_{N_Y}^*, b_{N_Y}^*]_Y\}$$
$$P_U^* = \{[a_0^*, b_0^*]_U, \ldots, [a_{N_U}^*, b_{N_U}^*]_U\}$$
$$P_X^* = \{[a_0^*, b_0^*]_X, \ldots, [a_{N_X}^*, b_{N_X}^*]_X\} \quad (2.10)$$

Якщо:

$$\forall X \in P_X^*, \forall U \in P_U^* : U_\varepsilon(F_\varepsilon^S(X_i, U_i)) \cap P_Y^* \neq 0$$
$$\forall Y \in P_Y^* : \exists X \in P_X^*, U \in P_U : F_\varepsilon^S(X, Y) \subset U_\varepsilon(Y), \quad (2.11)$$

Нейронні мережі - панацея апроксимації Цибенко



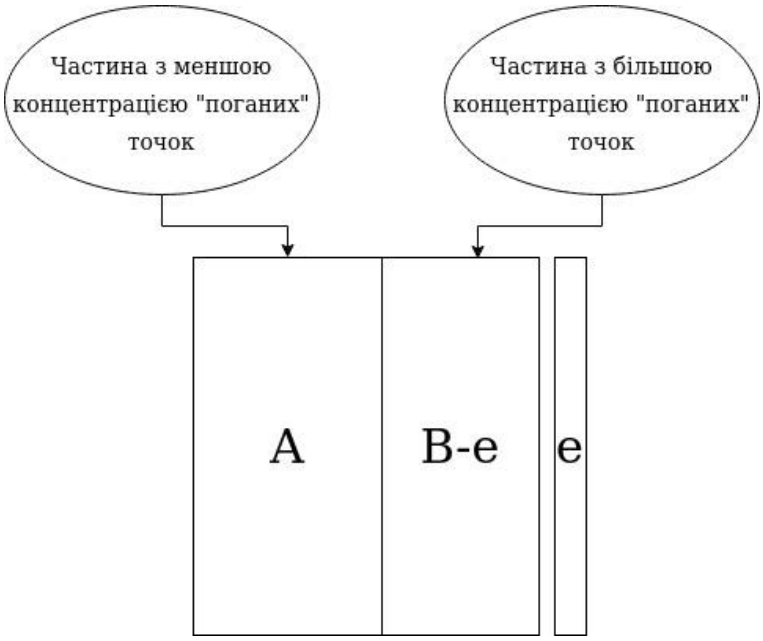
Нейронні мережі - панацея апроксимації Цибенко

Нехай задано деяку сигмоідальну функцію, наприклад $\varphi(\xi) = \frac{1}{1+e^{-x}}$. Тоді якщо $f : X \to R$, ($X \subset R^n$- будь яка компактна множина) будь-яка неперервна функція дійсних аргументів, то для будь якої точності $\varepsilon > 0$ існують вектори $w = w_1, \ldots, w_N$, $\alpha = \alpha_1, \ldots, \alpha_N$, $\Theta = \Theta_1, \ldots, \Theta_N$ та параметризована функція $G(*, w, \alpha, \Theta) : X \to R$ така, що:

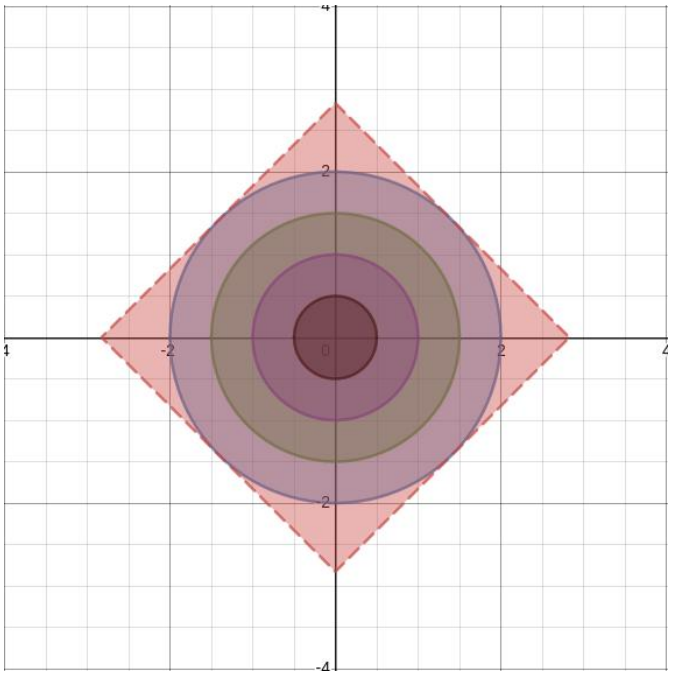
$$\forall x \in X : |G(x, w, \alpha, \Theta) - f(x)| < \varepsilon \tag{2.17}$$

де $G(x, w, \alpha, \Theta) = \sum_{i=0}^{N} \alpha_i \varphi(w_i x + \Theta_i), w_i \in R_n, \alpha, \Theta \in R^N$.

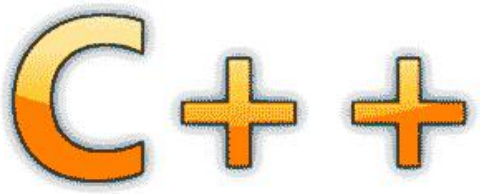
“Potato”-метод, або метод відсікання кінців



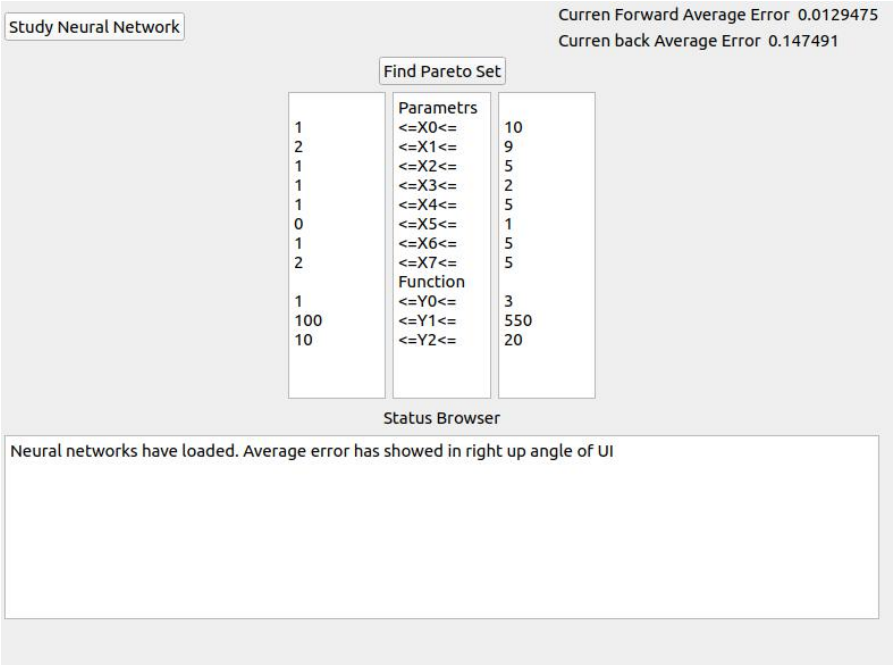
Грубий перебір не єдиний метод



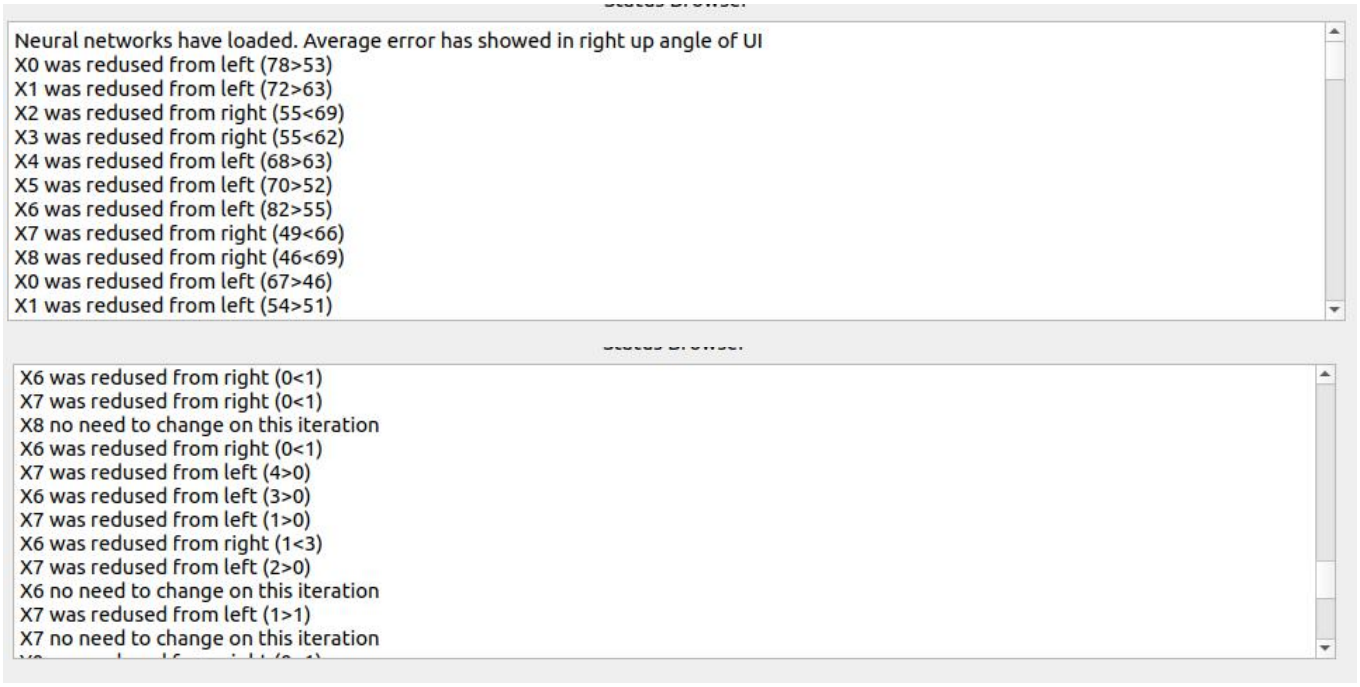
Вибір мови програмування для реалізації



Вибір мови програмування для реалізації



Результати роботи програми



Результати роботи програми

```
X0 was redused from left (11>8)
X1 was redused from left (8>8)
X2 was redused from left (16>6)
X3 was redused from left (11>3)
X4 was redused from right (5<10)
X5 was redused from left (17>3)
X6 was redused from right (4<11)
X7 was redused from left (7>3)
X8 was redused from right (4<7)
X0 was redused from right (5<8)
X1 was redused from right (7<13)
X2 was redused from left (9>6)
```

```
X6 no need to change on this iteration
X7 no need to change on this iteration
X8 was redused from right (0<1)
X0 no need to change on this iteration
X1 was redused from right (1<2)
X2 was redused from left (2>0)
X4 no need to change on this iteration
X8 no need to change on this iteration
X1 was redused from left (1>0)
X2 no need to change on this iteration
X1 was redused from left (1>0)
X1 no need to change on this iteration
```

Результати роботи програми

```
X6 was redused from right (1<3)
X7 was redused from left (2>0)
X6 no need to change on this iteration
X7 was redused from left (1>1)
X7 no need to change on this iteration
Y0 was redused from right (0<1)
Y1 was redused from left (1>0)
Y0 no need to change on this iteration
Y1 was redused from left (1>1)
Y1 was redused from right (0<1)
Y1 was redused from left (1>0)
Y1 no need to change on this iteration
```

```
X2 no need to change on this iteration
X1 was redused from left (1>0)
X1 no need to change on this iteration
Y0 was redused from right (0<2)
Y1 was redused from right (0<1)
Y0 was redused from left (1>0)
Y1 was redused from left (1>0)
Y0 was redused from left (1>0)
Y1 no need to change on this iteration
Y0 was redused from right (0<1)
Y0 no need to change on this iteration
```

Результати роботи програми

```
          Parametrs
5.16592 <=X0<=    8.2057
4.41879 <=X1<=    6.41174
2.07961 <=X2<=    3.44427
1.53552 <=X3<=    1.88013
2.57575 <=X4<=    3.7613
0.55788 <=X5<=    0.854266
2.48171 <=X6<=    3.62054
3.23111 <=X7<=    4.09386
1.51917 <=X8<=    2.13009
          Functions
108.777 <=Y0<=    536.722
10.3911 <=Y1<=    19.8059
```

Висновки до роботи

Дякую за увагу!