МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ХРИСТОФОР АНДРІЙ ОЛЕГОВИЧ

Допускається до захисту:
завідувач кафедри інформаційних технологій, канд. техн. наук, доцент
_____О.В.Зелінська
«__»_____2024р.

ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ ЗА ДОПОМОГОЮ НЕЧІТКИХ КОГНІТИВНИХ КАРТ

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (магістерська) робота

Науковий керівник:
Ротштейн О. П.,
професор кафедри
інформаційних технологій,
докт. техн. наук, професор
_____
(підпис)

Оцінка : ___/ ___ / _____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця 2024

АНОТАЦІЯ

Христофор А.О. Прогнозування часових рядів за допомогою нечітких когнітивних карт.

Спеціальність 122 «Комп’ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

Головною метою виконання даної кваліфікаційної роботи є розробка нового підходу до прогнозування часових рядів за допомогою нечітких когнітивних карт.

У вступі акцентовано увагу на актуальності теми прогнозування часових рядів та знаходження нового методу для цього.

Перший розділ присвячений аналізу предметної області та літератури на тему нечітких когнітивних карт, прогнозування часових рядів та теорії хаосу.

Другий розділ присвячений методології прогнозування часових рядів за допомогою нечітких когнітивних карт та алгоритму виділення нечіткої когнітивної карти з часового ряду.

У третьому розділі проаналізовано отримані результати, а також проведено порівняння з іншими наявними методами.

Ключові терміни: нечітка когнітивна карта, НКК, часовий ряд, фазовий портрет.

ABSTRACT

Khrystofor A.O. Time series forecasting using fuzzy cognitive maps.

Speciality 122 «Computer Science». Vasyl' Stus Donetsk National University, Vinnytsia, 2024.

The primary objective of this qualification paper is to develop a novel approach to time series forecasting using fuzzy cognitive maps.

The introduction emphasizes the relevance of time series forecasting and the need for a new method to address this task.

The first chapter is dedicated to an analysis of the subject area and a review of the literature on fuzzy cognitive maps, time series forecasting, and chaos theory.

The second chapter focuses on the methodology of time series forecasting using fuzzy cognitive maps and the algorithm for extracting a fuzzy cognitive map from a time series.

The third chapter analyzes the obtained results and conducts a comparison with other existing methods.

Key terms: fuzzy cognitive map, FCM, time series, phase portrait.

ЗМІСТ

ВСТУП 5

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ НЕЧІТКИХ КОГНІТИВНИХ КАРТ, ЧАСОВИХ РЯДІВ ТА ТЕОРІЇ ХАОСУ 7

1.1 Нечіткі когнітивні карти 7
1.2 Алгоритм навчання 11
1.3 Передбачення часових рядів 16
1.4 Основи теорії хаосу 18

РОЗДІЛ 2. МЕТОДОЛОГІЯ ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ ЗА ДОПОМОГОЮ НЕЧІТКИХ КОГНІТИВНИХ КАРТ 20

2.1 Створення нечіткої моделі часового ряду 20
2.2 Моделювання нечіткої когнітивної карти 21
2.3 Алгоритм прогнозування часового ряду за допомогою НКК 23
2.4 Відновлення фазового портрету 24
2.5 Оптимальна кількість вузлів та оцінка точності 26
2.6 Порівняння з іншими методами прогнозування 28
2.7 Стек технологій та інструменти для реалізації моделі 30
2.8 Вибір даних для дослідження 32

РОЗДІЛ 3. КОМП’ЮТЕРНИЙ ЕКСПЕРИМЕНТ 35

3.1 Атмосферний тиск 35
3.2 Споживання електроенергії 39
3.3 Втрати ворога 43
3.4 Порівняння з іншими методами прогнозування 47

ВИСНОВКИ 48

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ 50

ДОДАТОК А 54

ВСТУП

Прогнозування часових рядів є важливою задачею в багатьох галузях, таких як економіка, фінанси, медицина, інженерія, природничі науки та багато інших. Завдання полягає в тому, щоб на основі даних про минулі значення часового ряду передбачити його майбутні значення. Точні прогнози часових рядів можуть бути використані для прийняття кращих рішень, запобігання ризикам та оптимізації процесів.

Існує багато методів прогнозування часових рядів, але не всі вони однаково ефективні. Традиційні статистичні методи, такі як авторегресивні інтегровані ковзаючі середні (ARIMA), часто не можуть впоратися з нелінійними та складними часовими рядами. З іншого боку, штучні нейронні мережі та інші алгоритми машинного навчання, хоча й можуть бути більш гнучкими, часто вимагають значних обчислювальних ресурсів та складних налаштувань.

Нечіткі когнітивні карти (НКК) є альтернативним методом прогнозування часових рядів, який набуває все більшої популярності. НКК - це динамічні моделі, які представляють знання про систему у вигляді графа, де вузли представляють концепції або змінні, а зв'язки між ними представляють причинно-наслідкові зв'язки. НКК можуть моделювати нелінійні та складні взаємозв'язки між змінними, а також враховувати невизначеність та нечітку інформацію.

Об'єктом дослідження є методи прогнозування часових рядів за допомогою нечітких когнітивних карт. Предметом дослідження є розробка нового методу прогнозування часових рядів на основі НКК, який можна буде використовувати для рядів з різними характеристиками.

Мета дослідження - розробити новий метод прогнозування часових рядів на основі нечітких когнітивних карт, який буде:

- точним: метод буде генерувати точні прогнози часових рядів;
- ефективним: метод буде працювати швидко та ефективно;
- гнучким: метод можна буде використовувати для прогнозування часових рядів з різними характеристиками.

Наукова новизна дослідження полягає в розробці нового методу прогнозування часових рядів на основі НКК, який буде більш точним, ніж інші методи прогнозування, і його можна буде використовувати для прогнозування часових рядів з різними характеристиками.

Результатом роботи є розглянута література на тему нечітких когнітивних карт, часових рядів та теорії хаосу, наведено теоретичний частину методології прогнозування часових рядів за допомогою НКК, а також наведено комп’ютерну програму, яка реалізує новий метод прогнозування часових рядів, що поєднує в собі переваги нечіткої логіки та теорії динамічних систем.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ НЕЧІТКИХ КОГНІТИВНИХ КАРТ, ЧАСОВИХ РЯДІВ ТА ТЕОРІЇ ХАОСУ

У цьому розділі буде представлено огляд нечітких когнітивних карт та їх застосування до прогнозування часових рядів. Спочатку буде описано структуру та алгоритми навчання НКК. Потім буде розглянуто, як НКК можна використовувати для прогнозування майбутніх значень змінних. Нарешті, буде представлено огляд досліджень, які демонструють ефективність НКК для прогнозування часових рядів.

1.1 Нечіткі когнітивні карти

Нечіткі когнітивні карти(НКК) були запропоновані Бартом Коско [1] як особливе сімейство когнітивних карт. Термін Когнітивна Карта був введений дослідником та політологом Робертом Аксельродом [2] при дослідженні когнітивної активності щурів, який проводив експерименти над їх вибором шляху до їжі. Пізніше він використав знання політології для формування когнітивних карт у вигляді причинно-наслідкових зв’язків для прийняття рішень у політиці.

Когнітивна карта – причинно-наслідкова модель, що складається з понять(концептів) та причинно-наслідкових зв’язків між ними. З математичної точки зору когнітивна карта – це орієнтований граф, вершинами якого є концепти, а ваги дуг відображають силу впливу зміни одного поняття на зміну іншого. Термін “когнітивна” означає, що даними для моделювання слугують суб’єктивні думки експерта, які можна інтерпретувати як “збільшується” або “зменшується”. Наприклад, “через збільшення першого параметру, збільшується другий параметр”.

Нечітка когнітивна карта – це гібридна техніка, заснована на концепції когнітивних карт у поєднанні з нечіткою логікою для представлення невизначеності та складних характеристик системи. Це стало потужною парадигмою для представлення знань, надаючи гнучкий механізм для

представлення знань інтелектуальних систем. У НКК існують часткові зв’язки між концептами, які можуть бути представлені як нечіткі підмножини, що вказують нечіткі значення в проміжку [-1,1]. Ці значення відповідатимуть поняттям “слабко”, “середнє”, “сильно”.

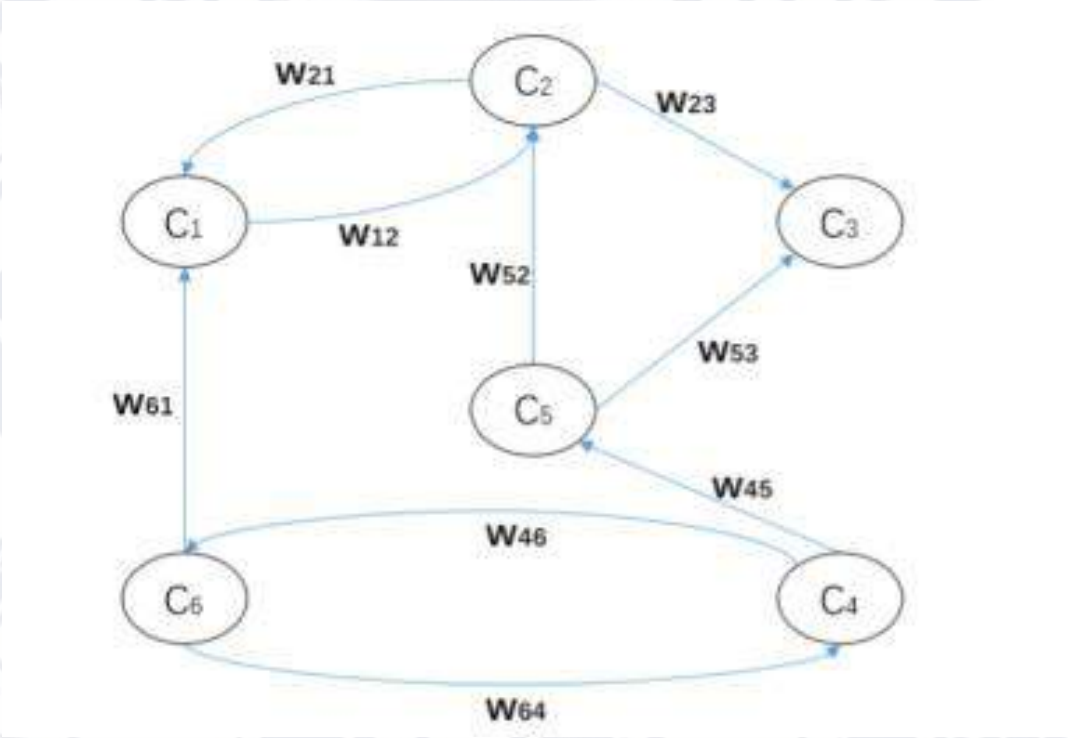

Рис. 1.1 - Приклад нечіткої когнітивної карти [3]

Для кожної нечіткої когнітивної карти з k концептів, причинно-наслідкові зв’язки між концептами можна представити у вигляді квадратної матриці ваг W розміром k*k.

$$W = \begin{pmatrix} w_{11} & \cdots & w_{1k} \\ \vdots & \ddots & \vdots \\ w_{k1} & \cdots & w_{kk} \end{pmatrix}$$

Кожен елемент w_{ij} матриці показує направлений зв’язок між концептами c_i та c_j і його значення показує ступінь впливу двох понять один на одного. Чим вище абсолютне значення ваги понять, тим сильніший зв’язок між парою понять. Існує три можливі варіанти для кожного окремого вагової матриці:

1. $w_{ij} < 0$: означає від'ємне відношення між концептами c_i та c_j, тобто зі збільшенням/зменшенням значення вузла c_i, значення c_j зменшується/збільшується;
2. $w_{ij} < 0$: означає додатне відношення між концептами c_i та c_j, тобто зі збільшенням/зменшенням значення вузла c_i, значення c_j збільшується/зменшується;
3. $w_{ij} = 0$: означає нульове відношення між концептами c_i та c_j.

Загалом, нечітку когнітивну карту визначає кортеж з 4 елементів: (C, W,a,f), де:

- C – набір із k концептів $[c_1 \ldots c_k]$, які є змінними(вузлами графа), що складають систему;
- W – матриця зв'язків між концептами;
- a – вектор стану $(a_1 \ldots a_k)$, що містить набір усіх концептів;
- f – функція активації, залежність значення концепту c_i на кроці (t+1) від значень концептів c_j(j=1, 2, ..., k) на кроці t. У будь-який момент часу t кожен концепт має значення активації або нечітке значення в інтервалі [0,1]. Операція НКК враховує загальний вплив концептів, напряму з'єднаних з поточним на кожній ітерації.

Загалом, виходячи з літератури [28, 15], існують різні види правил зв'язку з минулими значеннями для кожного концепту. Рівняння (2) вказує на правило переходу Коско для концепції c_i в момент часу t+1 без пам'яті, яке широко використовується в багатьох алгоритмах на основі нечітких когнітивних карт.

$$a_i(t+1) = f\left(\sum_{j=1, j\neq i}^{k} w_{ij} a_i(t)\right) (1)$$

Після того, як задано усі значення вагових коефіцієнтів НКК(тобто матриці взаємозв'язків), НКК починає із заданого початкового вектора стану і виконує обчислення послідовних ітерацій відповідно до рівняння (1). На кожному кроці НКК генерує вектор стану a(t) = (a_i(t), . . ., a_k(t), який містить значення усіх

концептів у момент часу t. Це правило оновлення ітеративно повторюється, доки не буде виконано умови завершення. Це означає, що в будь-якій ітерації отримується нове значення стану концептів і після певної кількості повторень для нечіткої когнітивної карти може виникнути один із трьох можливих станів [16]: фіксовану точку рівноваги, обмежений цикл і хаотичну поведінку. В обох випадках хаотичного чи циклічного результат може бути частково ненадійним через відсутність стабільності. Коли НКК досягає фіксованої точки рівноваги, можна зробити висновок, що НКК збіглась.

Хоча функція (1) використовується в багатьох алгоритмах на основі НКК, в літературі зустрічаються і інші її модифікації [4, 5]. Так, існує модифіковане правило оновлення, в якому крім ваг взаємозв'язків і значень активації інших концептів, для кожного вузла також враховується і його власне попереднє значення, яке називається фактором пам'яті.

$$a_i(t+1) = f\left(a_i(t) + \sum_{j=1, j\neq i}^{k} w_{ij} a_i(t)\right) (2)$$

Проте, наведена функція (2) не враховує вплив зв'язку з самим собою, тож існують і інші правила, в яких додано "самовплив".

$$a_i(t+1) = f\left(a_i(t) + \sum_{j=1}^{k} w_{ij} a_i(t)\right) (3)$$

Найбільш поширені функції активації:

- двійкова: $f(x) = \begin{cases} 0, \text{якщо} x < 0 \\ 1, \text{якщо} x \geq 0 \end{cases}$
- трійкова: $f(x) = \begin{cases} 1, \text{якщо} x \geq 0.5 \\ 0, \text{якщо} -0.5 < x < 0.5 \\ -1, \text{якщо} x \leq -0.5 \end{cases}$
- сигмоїдна: $f(x) = \frac{1}{e^{-x}+1}$

Вибір правильної функції активації для моделювання динамічної поведінки системи залежить від предметної області та чіткого розуміння змодельованої системи. Двійкова і трійкова функції є дискретними і породжують скінченну

кількість станів, а сигмоїдна функція належить до групи безперервних, тобто створює нескінченні стани. Безперервні функції використовуються для моделювання більш якісних та складних випадків. Так, дослідження [6, 7] показали більшу ефективність сигмоїдної функції активації для нечітких когнітивних карт.

1.2 Алгоритм навчання

Для отримання більш точних результатів при використанні нечітких когнітивних карт, застосовуються методи їх навчання. Навчання НКК полягає у тому, щоб отримати вагову матрицю відповідно до експертного втручання або вилучити її з наявних історичних даних. В літературі зустрічається багато методів навчання, але здебільшого вони поділені на 3 категорії на основі базових підходів. Розглянемо їх більш детально [8].

Методи на основі теорії Гебба. Основною метою даних методів є знаходження вагової матриці на основі знань експертів у галузі. Першим алгоритмом, заснованим на цій теорії, є диференціальне навчання Геббі. Був представлений Дікерсоном і Коско [9]. У даному алгоритмі вагові матриці змінюються, коли змінюється значення відповідних концептів. Таким чином, значення ваг оновлюється неодноразово, але якщо значення понять не змінюються, значення ваг залишаються незмінними в наступній ітерації. Процес навчання змінює значення ваги, поки не досягне бажаного стану. Основним недоліком цієї методики є те, що вага між кількома поняттями оновлюється шляхом урахування лише відповідних понять, а вплив інших понять ігнорується. Крім того, значну роль відіграє порядок представлення даних [10].

Щоб вирішити вищезазначену проблему, у було запропоновано вдосконалену версію методу, відому як збалансований диференціальний алгоритм [11]. У процесі оновлення ваги враховується зміна всіх концепцій на одному етапі та в одному напрямку. Цей метод покращує обмеження диференціального навчання Геббі, враховуючи значення всіх понять, які змінюються одночасно з

оновленням вагових коефіцієнтів. Проте, хоча даний метод зменшує обмеження попереднього, його застосування обмежене лише бінарними нечіткими когнітивними картами.

У 2004 році було представлено два інші алгоритми навчання на базі теорії Гебба, які називаються Active Hebbian Learning і Nonlinear Hebbian Learning [12, 13]. У методі NHL діапазон значень понять, а також знак ребер уточнюються експертним втручанням, а нульові ребра не оновлюються. Отже, основним недоліком NHL є рекомендована фахівцями побудова початкового графа. Початкова структура графа, отримана від експертів, зберігається в процесі навчання, таким чином, зберігається його фізична інтерпретація. Критерій зупинки формується на основі обмежень, накладених на вузли. Вагові коефіцієнти коригуються, коли задовольняються критерії зупинки, які включають: досягнуто достатньо близького рішення до бажаної реакції або ідентифіковано атрактор із фіксованою точкою.

Тож, дана група методів є корисною завдяки низьким затратам на обчислення, простоті використання, збереженню причинно-наслідкових зв’язків скоригованих ваг. Проте у даних методів є і свої недоліки, такі як погане узагальнення, залежність від експертів, залежність від початкових станів тощо. Можна зробити висновок, що ця група алгоритмів придатні до задач керування [14].

Навчання на основі популяцій. У цій групі методів навчання відбувається на основі історичних даних, які використовуються у формі пар введення-виведення для навчання моделі. Іншими словами, ми розробляємо алгоритм із правильною відповіддю для кожного елемента набору даних. У таких задачах алгоритм намагається обчислити вихід для кожного нового входу, враховуючи набори даних. Тому основною метою є отримання вагової матриці для відображення впливу між концепціями шляхом використання алгоритмів оптимізації для мінімізації розбіжності між цільовими та прогнозованими відповідями.

В роботі представлено метод Particle Swarm Optimization(PSO), який використовує історичні дані для формування найбільш-можливо оптимальної матриці ваг для нечіткої когнітивної карти [15]. Автори використали запропоновані методи для мінімізації функції активації для бажаного кінцевого значення НКК з фіксованою архітектурою.

Real-Coded Genetic Algorithm(RCGA) був використаний в роботі як метод навчання НКК для створення її структури на основі історичних даних в рамках часових рядів, які включають єдину послідовність значень вектора стану, без втручання людини [16]. Запропонований RCGA використовується для мінімізації трьох різних функцій активації для вибору найкращої. Потім обрана функція використовується для проведення експериментів з різною кількістю концептів та щільністю НКК. Чим більший обсяг вхідних даних, тим точніше навчання. З іншого боку, точність залежить від розміру вхідних даних і ефективність навчання погіршується при збільшенні розміру карт.

Multi-Objective Evolutionary – інший еволюційний алгоритм навчання, запропонований в на основі генетичних алгоритмів для підтримки багатоцільових задач прийняття рішень [16]. Цей метод застосовується для пошуку оптимальної вагової матриці, яка задовольняє попередньо визначеним рівням активації серед вузлів-учасників, шляхом випадкового вибору початкових ваг.

Evolution strategy (ES) була представлена в як ефективний спосіб проектування та побудови нечітких когнітивних карт [17]. В роботі зосереджено увагу на надійність та гнучкість процедури навчання, яка відкидає вплив зовнішнього втручання для тонкого налаштування параметрів НКК, особливо в складних системах. Запропонований метод, який є комбінацією НКК та ES, розглядається на предмет потенційної реалізації в системах на основі нечітких когнітивних карт. У цьому алгоритмі процес навчання зупиняється, коли отримано оптимальну структуру НКК.

Memetic Particle Swarm Optimisation (MPSO) — метод навчання НКК для вилучення матриці ваг шляхом мінімізації функції активації для побудови бажаної системи. Запропонований MPSO є комбінацією PSO як алгоритму

глобального пошуку та алгоритму Гука і Дживса (HJ) як компонента локального пошуку. Остаточні результати довели перевагу MPSO у порівнянні з PSO.

Алгоритм Simulated Annealing (SA) було представлено як ще один метаевристичний метод навчання НКК для вилучення вагових матриць з вхідних історичних даних без втручання експертів. Порівняння продуктивності SA та GA ілюструє, що при більшій кількості вузлів (складні НКК) алгоритм GA погіршує продуктивність НКК, в той час як SA покращує якість навчання, покриваючи обмеження GA, а також покращує швидкість навчання для кожної кількості вузлів. Таким чином, запропонований метод ефективно працює для будь-якого розміру карти. Тобто, у цьому експерименті метод навчання GA використовується для малих розмірів КДК, а SA - для великих розмірів КДК.

Крім наведених вище, в літературі представлено і багато інших алгоритмів навчання нечітких когнітивних карт на основі популяцій. Так, наприклад, представлено алгоритми Asexual Reproduction Optimization (ARO) та Modified ARO(MARO), Immune Algorithm(IA), Ant Colony Optimization (ACO) та інші [10, 18, 19].

Тож, методи навчання на основі популяцій широко використовуються завдяки меншій похибці, вищій функціональності, надійності та здатності до узагальнення. З іншого боку, загальні витрати часу, велика кількість параметрів і процесів навчання, можлива недоступність історичних даних є суттєвими недоліками таких алгоритмів. Тож, ця група методів більш широко застосовується у прогнозуванні часових рядів, задачах класифікації, моделюванні віртуальних систем тощо.

Гібридні методи. Обидва розглянуті підходи обмежені в застосуванні у деяких випадках. Щоб вирішити недоліки, що існують, і покращити ефективність, були запропоновані гібридні методи, які включають геббівський та популяційний методи.

У був представлений гібридний метод, який є сумішшю методів неконтрольованого навчання, алгоритму NHL і стратегії диференційної еволюції (DE) для усунення деяких недоліків нечітких когнітивних карт, динамічного

покращення поведінки і підвищення гнучкості НКК [20]. Простота реалізації, недорогі обчислення, менша кількість контрольних параметрів, а також ефективне поводження з нелінійними, недиференційованими та мультимодальними функціями відповідності є основними перевагами досліджуваної методики в цій статті. Запропонований алгоритм NHL-DE розділений на два кроки. В першу чергу метод NHL використовується для вивчення НКК, потім DE використовується для перенавчання НКК. Метою першого етапу є пошук відповідних вагових коефіцієнтів, тоді як відповідальністю другого етапу є перерахунок вагових коефіцієнтів і мінімізація оптимальної фітнес-функції. Відзначаючи, що в цьому методі початкова популяція DE безпосередньо залежить від продуктивності першого етапу, який витягує апріорні знання для включення в еволюційні обчислення, базується на хороших рішеннях на першому етапі. У цьому методі розглядаються дві умови завершення, і процес мінімізації буде завершено, коли критерій оптимізації задовольняється на другому етапі. Експериментальні результати ілюструють високу швидкість і ефективність моделі, розглянутої в трьох різних моделях НКК.

Інші методи. Існують й інші методи, запропоновані дослідниками для побудови або оптимізації систем на основі нечітких когнітивних карт, які не належать до трьох вищезгаданих категорій.

В роботі Аміта Конара розроблено новий метод неконтрольованого навчання. Ваги спрямованих ребер адаптуються від переходу до мережі Петрі через процес навчання. Для аналізу динамічної поведінки алгоритму було використано алгоритм навчання на основі Геббіана з урахуванням природного спаду ваг. Після збіжності алгоритму навчання мережа може бути використана для обчислення вірогідності бажаних пропозицій з поданих вірогідностей аксіом (місць, де немає вхідних дуг). Умовний характер алгоритму з точки зору стабільності дозволяє використовувати модель у складних процесах прийняття рішень та навчання. Інша модель була розроблена для уточнення знань шляхом адаптації ваг у нечіткій мережі Петрі з використанням іншої форми геббівського навчання. Крім того, в роботі було розроблено комбінований підхід нечітких

когнітивних карт та мереж Петрі для управління енергією автономних мікромереж з полігенерацією. Мережа Петрі використовується як активатор у структурі нечіткої когнітивної карти, що дозволяє активувати різні НКК в залежності від стану мікромереж.

1.3 Передбачення часових рядів

Часовий ряд - це послідовність даних, індексованих у хронологічному порядку. Це означає, що кожне значення даних у ряді має пов'язану з ним часову мітку, яка вказує, коли воно було виміряно або спостережено.

Часові ряди використовуються в багатьох галузях, включаючи економіку, фінанси, метеорологію, інженерію та охорону здоров'я. Їх можна використовувати для:

- виявлення закономірностей і трендів — аналіз часових рядів може допомогти виявити закономірності та тренди в даних, які інакше було б важко побачити. Це може бути корисно для прогнозування майбутніх значень, розуміння причин минулих подій або прийняття кращих рішень;
- розуміння причинно-наслідкових зв'язків — часові ряди можуть бути використані для вивчення причинно-наслідкових зв'язків між різними змінними. Це може бути корисно для розуміння того, як різні фактори впливають на певний результат;
- прогнозування.

Прогнозування часових рядів – передбачення майбутніх значень певної величини на основі моделі, навченої на історичних спостереженнях. Іншими словами, нехай Y(t) є заданим часовим рядом. Метою моделей прогнозування часових рядів є прогнозування наступних значень часових рядів з урахуванням горизонту прогнозування (H), який можна розділити на чотири категорії, включаючи дуже короткострокове, короткострокове, середньострокове та довгострокове прогнозування.

Однак ідеальної моделі для прогнозування точних майбутніх значень не існує через наявність невизначеності та нелінійності, пов’язаної з більшістю явищ реального світу. Відповідно, у літературі представлена велика кількість методів прогнозування часових рядів для виконання операції прогнозування та оцінки прогнозованого значення, від статистичних методів (наприклад, ARMA, ARIMA) до деяких нових інтелектуальні методи (наприклад, LSTM, CNN). Таким чином, побудова належної моделі є досить важливою задачею для знаходження точних прогнозованих значень порівняно з реальними.

Протягом останнього часу дослідження застосування нечітких когнітивних карт у прогнозуванні часових рядів показали значний прогрес, завдяки своїй простоті, точності та масштабованості. Суть гібридних методів полягає в оновленні матриць зв’язків, отриманих з історичних даних, на основі знань експертів. Хоча гібридні алгоритми можуть добре справлятися зі складними системами, згідно з літературою, цьому методу присвячено мало досліджень.

Модель прогнозування M в нечіткій когнітивній карті пов'язана з мережею на основі НКК. Це означає, що після зіставлення нейронів НКК зі змінними часового ряду, навчання НКК розглядається для вилучення ваг шляхом тестування в межах горизонту прогнозування. Основною проблемою в моделях прогнозування часових рядів на основі НКК є формулювання її структури, включаючи нейрони та причинно-наслідкові зв'язки між парами нейронів, на основі заданого часового ряду. У часових рядах, як рядах числових даних, ідентифікація понять не така проста, як у традиційних системах, і потребує дослідження. З цієї причини різні дослідження з прогнозування часових рядів за допомогою НКК були зосереджені в основному на виявленні відповідної структури НКК, а також на визначенні найкращого алгоритму навчання для обчислення вагових матриць, що описують динамічну поведінку системи. Простіше кажучи, прогнозування часових рядів за допомогою НКК складається з двох етапів. По-перше, проектування структури НКК з використанням загальних успішних стратегій, включаючи гранулярність, представлення значень належності, нечітку кластеризацію с-середніх. Крім того, вейвлет-перетворення та

емпірична декомпозиція мод також пропонуються для ідентифікації структури НКК та підвищення ефективності прогнозування.

По-друге, навчання вагової матриці з використанням різних методів навчання, як було представлено раніше. Без високоякісних вагових матриць система втратить свою інтерпретованість, навіть за наявності кристально чистого міркування про нечітку когнітивну карту.

1.4 Основи теорії хаосу

Теорія хаосу – це галузь математики та фізики, яка вивчає системи, поведінка яких може бути надзвичайно чутливою до найменших змін у початкових умовах. Ця чутливість призводить до того, що навіть незначні відхилення можуть призвести до кардинально різних результатів з плином часу. Цей феномен часто називають "ефектом метелика": махання крилами метелика в одній частині світу може викликати шторм в іншій. Одним з найпростіших прикладів хаотичних систем є погода: атмосферні процеси є надзвичайно чутливими до початкових умов, що робить довгострокове прогнозування погоди складним завданням.

Ключові ідеї теорії хаосу:

- нелінійність: системи, які вивчає теорія хаосу, є нелінійними, тобто зміна однієї змінної не пропорційна зміні іншої.
- чутливість до початкових умов: навіть найменша зміна в початкових умовах може призвести до значно різних результатів у майбутньому.
- атрактори: це множини, до яких притягуються траєкторії системи з плином часу. Атрактори можуть бути простими (наприклад, точка) або дуже складними (фрактали).
- показник Ляпунова: це числова характеристика, яка вимірює швидкість розходження близьких траєкторій в фазовому просторі. Додатність показника Ляпунова зазвичай свідчить про хаотичну поведінку системи.

- фазовий портрет: це графічне зображення поведінки динамічної системи у фазовому просторі. Кожна точка на фазовому портреті відповідає одному конкретному стану системи, а лінія, що з'єднує ці точки, показує, як система еволюціонує з часом.

Як і в класичній теорії нелінійних коливань, теорія хаосу вивчає поведінку динамічних систем за допомогою фазового портрета. Особливістю класичної теорії коливань є те, що фазовий портрет динамічної системи будується на основі дослідження системи диференціальних рівнянь. На відміну від цього, теорія хаосу вивчає об'єкти, для яких система рівнянь невідома, а єдина доступна інформація про динамічний об'єкт – це часовий ряд. У такому випадку ми використовуємо часовий ряд для реконструкції штучного фазового портрета таким чином, щоб цей фазовий портрет мав ті самі топологічні властивості, що й вихідний фазовий портрет невідомої динамічної системи, яка генерувала спостережуваний часовий ряд[21].

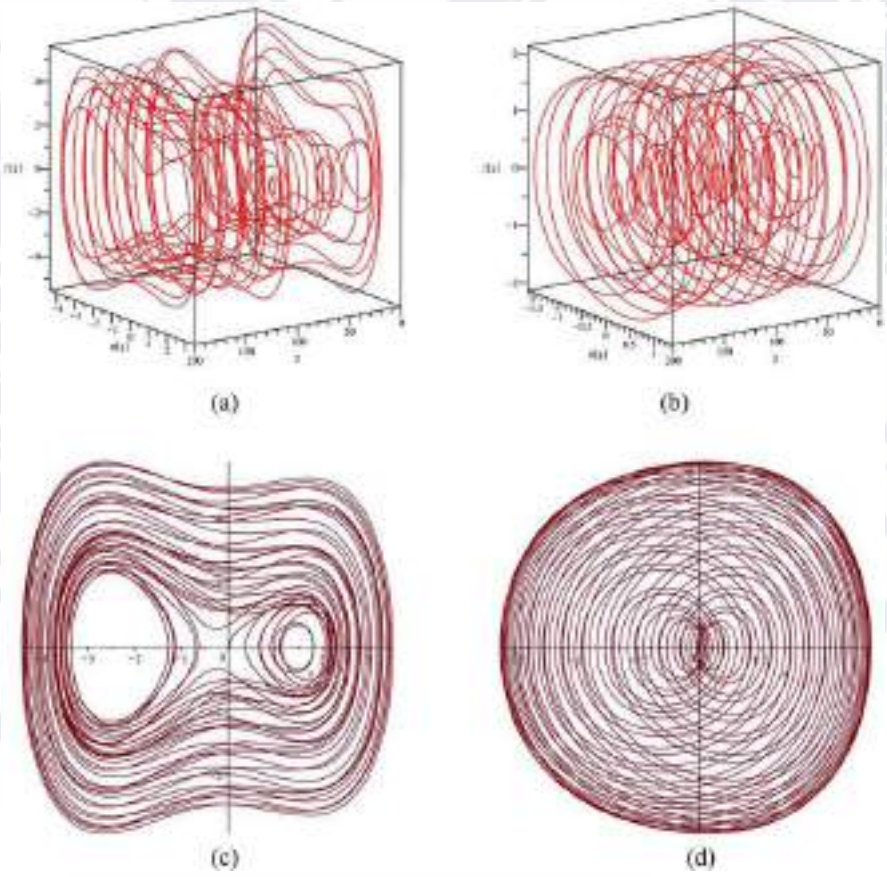

Рис. 1.2 - Приклади фазового портрету [22]

РОЗДІЛ 2. МЕТОДОЛОГІЯ ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ ЗА ДОПОМОГОЮ НЕЧІТКИХ КОГНІТИВНИХ КАРТ

У цьому розділі буде розглянуто один із методів прогнозування часових рядів за допомогою нечітких когнітивних карт. Цей процес можна поділити на наступні частини: створення нечіткої моделі часового ряду, дослідження нечіткої когнітивної карти, прогнозування та опрацювання отриманих результатів[23].

2.1 Створення нечіткої моделі часового ряду

На цьому етапі дані часових рядів перетворюються в нечіткі значення, що дозволяє краще відобразити невизначеність і варіативність даних.

Розглянемо часовий ряд $X = \{x_1, x_2 \dots x_m\}$, де m – кількість записів у часовому ряді. Трикутний нечіткий набір використовується для нечіткої обробки часового ряду X. Формула функції належності трикутного нечіткого набору виглядає наступним чином:

$$\mu(x; a; b; c) = \begin{cases} 0, x \le a \\ \dfrac{x-a}{b-a}, a \le x \le b \\ \dfrac{c-x}{c-b}, b \le x \le c \\ 0, x \ge c \end{cases}$$

де параметри a і c відповідають лівій та правій вершині трикутника, b — це середня вершина трикутника, і $a \le b \le c$.

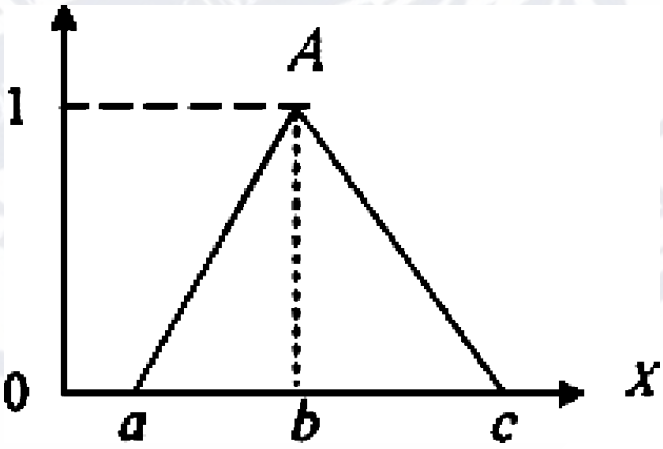

Рис. 2.1 – Візуальне зображення трикутної нечіткої множини [24].

Припустимо, що область визначення часового ряду X поділена на неперервні несуміжні інтервали за допомогою n точок $\{P_1, P_2 \dots P_n\}$, де кожна з цих точок поділу є нечіткою лінгвістичною змінною. У такому випадку ступінь належності вибіркових даних x_i до цих нечітких множин можна виразити як $\{\mu_1(x_i), \mu_2(x_i) \dots \mu_n(x_i)\}$.

Тоді, після того, як часовий ряд X буде нечітко описаний за допомогою трикутної нечіткої множини, матриця U, сформована з нечіткого часового ряду, матиме наступний вигляд:

$$U = \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mn} \end{bmatrix}$$

де m – кількість значень часового ряду, n – кількість поділених нечітких множин. Елементи кожного рядка матриці U є значеннями нечіткої належності відповідних вибіркових даних часового ряду, а сума елементів у кожному рядку дорівнює 1.

2.2 Моделювання нечіткої когнітивної карти

Отриманий у попередньому пункті нечіткий часовий ряд може бути використаний для навчання нечіткої когнітивної карти(НКК). Для цього, перепишемо рівняння (1) наступним чином:

$$a_j(t+1) = f\left(w_j a_i(t)\right) (4)$$

де $a(t) = [a_1(t), a_2(t), \dots a_n(t)]$ – значення станів усіх вузлів НКК у момент часу t, $a_j(t+1)$ – значення стану j-го вузла в момент часу t+1, а $w_j = [w_{1j}\ w_{2j} \dots w_{nj}]^T$ – j-й стовпець матриці ваг w. Як функцію активації використовуватимемо сигмоїдну функцію $f(x) = \frac{1}{1+e^{-\lambda x}}$. Тоді, після оберненого перетворення рівняння (4) маємо:

$$f^{-1}(a_j(t+1)) = w_j a_i(t) (5)$$

Отримане рівняння – лінійне, причому $f^{-1}(y) = -\frac{1}{\lambda}\ln\frac{1-y}{y}$ – функція, обернена до сигмоїдної функції f.

Розглянемо матрицю нечіткого часового ряду U. Нехай $Z = [U_1, U_2, \dots U_{m-1}]^T$, де $U_i = [u_{i1}\ u_{i2}\ \dots\ u_{in}], Y_j = f^{-1}([u_{2j}\ u_{3j}\ \dots\ u_{mj}]^T)$. Підставимо ці значення у рівняння (5). Маємо:

$$Y_j = ZW_j (6)$$

Z та W_j можна розглядати як незалежні та залежні змінні історичних даних відповідно. Тому, задачу навчання НКК можна перетворити на розв'язання задачі найменших квадратів. Наступна цільова функція побудована для розв'язання W_j:

$$\arg\min_{W_j} \left|ZW_j - Y_j\right|^2, \text{ з умовою } |W_j|_\infty \leq 1 (7)$$

Норму $\left|ZW_j - Y_j\right|^2$ використовують для мінімізації помилки найменших квадратів між фактичним значенням та прогнозованим значенням, щоб отримати розумне наближене рішення для W_j. Обмеження $|W_j|_\infty \leq 1$ забезпечує, що значення кожного елемента матриці ваг знаходиться в інтервалі [−1,1], де нескінченна $|W_j|_\infty = \max\{|w_{1j}|, |w_{2j}|, \dots, |w_{nj}|\}$. Вищезазначена формула є стандартною задачею конвексної оптимізації з лінійними обмеженнями[25].

Прогнозування часових рядів базується на історичних даних для виведення майбутніх даних. Вектор стану НКК у момент часу t задається як $U_t = [u_{t1}\ u_{t2}\ \dots\ u_{tn}]$. Коли матриця ваг W визначена, прогнозоване значення $\widehat{U}_{t+1}$ вектору стану в момент часу t+1 можна обчислити за наступною формулою:

$$\widehat{U}_{t+1} = f(U_t W) (8)$$

З її допомогою можна отримати вектор стану НКК в наступний момент часу, а отже можна і обчислити матрицю прогнозування $\widehat{U}$. Нарешті, щоб отримати кінцеві значення прогнозованого ряду, необхідно вектор стану перевести з когнітивної моделі у числові значення. Зробити це можна за допомогою наступної формули:

$$\hat{x}_{t+1} = \frac{\sum_{j=1}^{n} \hat{u}_{(t+1)j} P_j}{\sum_{j=1}^{n} \hat{u}_{(t+1)j}} (9)$$

де n – кількість вузлів НКК, P_j – трикутне нечітке лінгвістичне значення j-го вузла НКК.

2.3 Алгоритм прогнозування часового ряду за допомогою НКК

Для побудови нечіткої когнітивної карти використовуємо метод мультимодального навчання. Модель навчається на часових рядах, формуючи набір нечітких моделей. Об'єднання їхніх виходів дає кінцеве рішення. Алгоритм має наступні кроки:

1. За допомогою трикутної нечіткої множини з певною заданою кількістю множин створюється нечітка модель часового ряду. На рис 2.2 зображено її приклад. Матриця U має розмірність n*m, де n – кількість поділених нечітких множин(у прикладі n = 4), m – кількість значень тренувальної вибірки часового ряду.

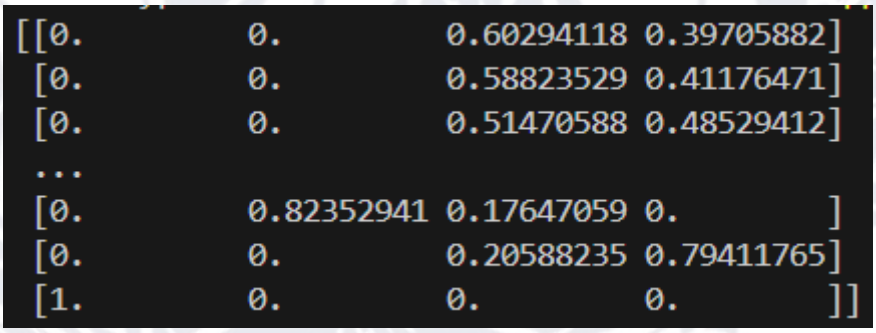

Рис 2.2 – Приклад нечіткої моделі часового ряду

2. Для створення набору нечітких моделей ми випадковим чином формуємо 100 підмножин матриці U однакової довжини. Кожна підмножина використовується для побудови окремої нечіткої когнітивної карти. Процес побудови кожної карти наступний:

- Вибирається підмножина матриці U.

- Для цієї підмножини розв'язується задача лінійної регресії (найменших квадратів) з метою знаходження оптимальних ваг зв'язків між вхідними та вихідними змінними нечіткої когнітивної карти.
- Отримані ваги визначають структуру нечіткої когнітивної карти.

3. Для об’єднання виходів усіх НКК знайдемо «вагу» кожної з них. Обчислимо середньоквадратичне відхилення реальних та прогнозованих кожною з карт даних. Тоді, чим менше відхилення дала конкретна когнітивна карта, тим більшу вагу вона має і тим більший її «вклад» повинен йти у загальний результат прогнозування. Вагу обчислюємо за формулою:

$$\omega_i = \frac{\frac{1}{rmse_i}}{\sum_{k=1}^{r} \frac{1}{rmse_k}}$$

де r – кількість когнітивних карт, які ми будували(в даному випадку r = 100).

Тоді, агрегованим результатом прогнозування усіх НКК буде:

$$y = \sum_{i=1}^{r} \omega_i * y_i$$

4. Перетворення прогнозованих значень із нечіткої моделі у значення часового ряду. За допомогою прогнозованого вектору належності та трикутної нечіткої множини, з якої створювалась нечітка модель, отримуємо прогнозовані дані часового ряду.

2.4 Відновлення фазового портрету

Реконструкція фазового простору в когнітивній науці частково базується на теоремі Такенса, яка каже про те, що часовий ряд одного сигналу системи містить інформацію про інші сигнали у всій системі[26]. Побудова фазового простору гарантує, що за певних умов ми покажемо (майже) точне зображення якісних і кількісних властивостей в оригінальному часовому ряді.

Для відновлення фазового портрету необхідно обрати або оцінити 2 ключові параметри: часову затримку τ (тау) та кількість вимірів d. Результатом такої реконструкції є матриця з d вимірів на різних множинних лагах τ, де d - кількість стовпців у кінцевій матриці вкладення. Неправильний вибір цих двох параметрів може призвести до спотворення властивостей і помилкового зазначення нелінійної системи, коли система насправді лінійна. Для оптимізації τ часової затримки та розмірності d використовується ряд статистичних інструментів, таких як (середня) взаємна інформація, автокореляція, помилкові найближчі сусіди (FNN), кореляції високого порядку, середнє зміщення (AD) тощо[27].

Оптимізація часової затримки: для знаходження оптимального значення τ потрібно знайти перший момент, коли траєкторії максимально розділені. Для цього використаємо функцію автокореляції та знайдемо перший локальний нуль.

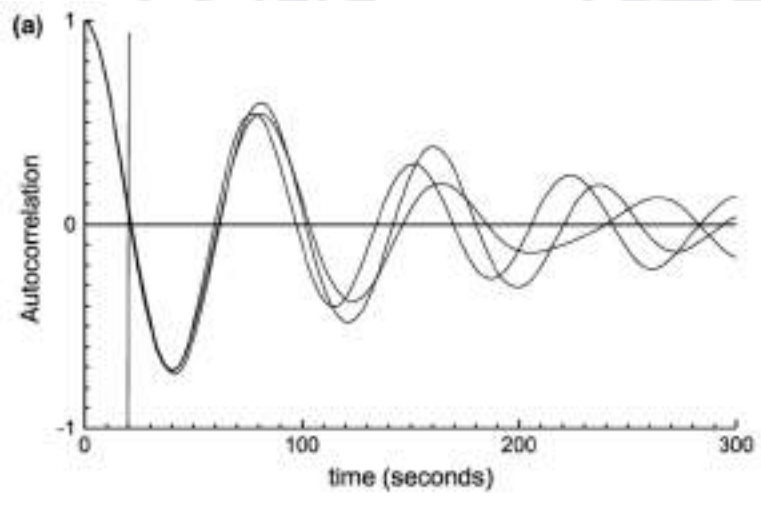

Рис 2.3 – Функції автокореляції декількох часових рядів[28]

Визначення оптимальної кількості вимірів: для цього використаємо метод найближчих хибних сусідів. Логіка оптимізації розмірності через алгоритм False Nearest Neighbor наступна: ми використовуємо поняття відстані на фазовій площині для обчислення оптимального значення d [29]. Відстань між двома точками даних є їх різницею в величині на площині. Вони є сусідами в реальному просторі, якщо ця відстань менше деякого порогу. Після вбудовування, якщо ця відстань значно змінилася, вони є хибними сусідами. Точно так само, якщо

відстань істотно не змінилася, то вони справжні сусіди. Вкладення виконується кілька разів для збільшення розмірів d. Значення, після якого кількість помилкових сусідів перестає змінюватися (значно), є оптимальною розмірністю.

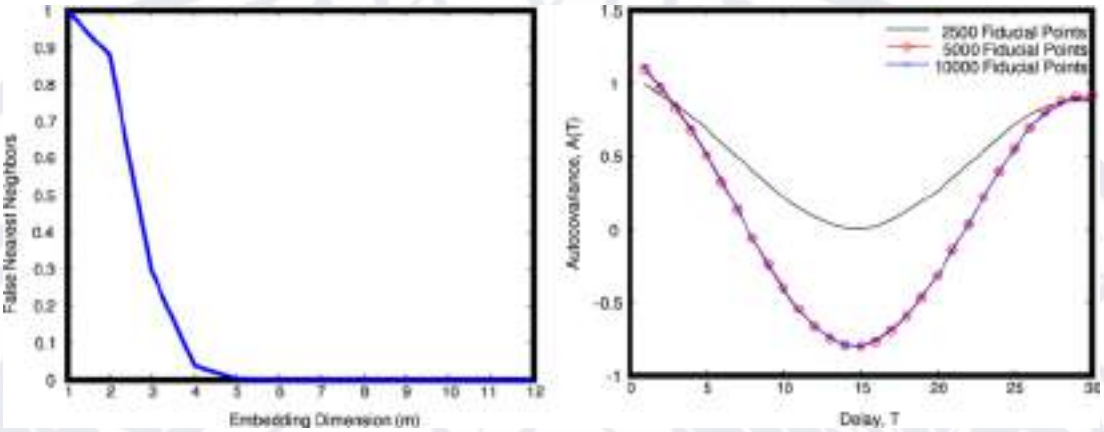

Рис 2.4 – Функції False Nearest Neighbor та функція автокореляції[30]

2.5 Оптимальна кількість вузлів та оцінка точності

Середньоквадратична помилка (RMSE) використовується для оцінки прогнозної продуктивності моделі:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(x(t) - y(t))^2}$$

де $x(t)$ позначає фактичне значення в момент часу t, а $y(t)$ представляє змодельоване значення в момент часу t, N – кількість даних. Очевидно, що чим менше значення RMSE, тим краща прогнозна продуктивність моделі.

Кількість концептів нечіткої когнітивної карти дуже сильно впливає на її продуктивність. Оптимальна кількість вузлів може бути встановлена під час аналізу помилки прогнозування RMSE. На рисунку 2.2 зображено кілька прикладів залежності RMSE від кількості параметрів(від 2 до 20). Аналіз цих даних показує, що зі збільшенням кількості концептів значення RMSE спочатку суттєво зменшується, але потім починає зростати або навіть різко зростати, коли кількість концептів перевищує певний поріг. Іншими словами, точність прогнозування не завжди покращується зі збільшенням кількості концептів.

Наприклад, для часового ряду Oldman оптимальним значенням є 10. У той же час, для часового рядуAnnual water після восьмого концепту суттєвого покращення точності реконструкції не спостерігається або вона навіть погіршується. Топологія моделі НКК ускладнюється зі збільшенням кількості концептів. Таким чином, оптимальне значення параметра обирається з урахуванням як складності топології моделі FCM, так і точності прогнозування.

У даній роботі кількість концептів нечіткої когнітивної карти будемо визначати на основі теорії хаосу. Оптимальною кількістю вважається мінімальна розмірність атрактора, починаючи з якої фазова траєкторія позбавляється самоперетинів.

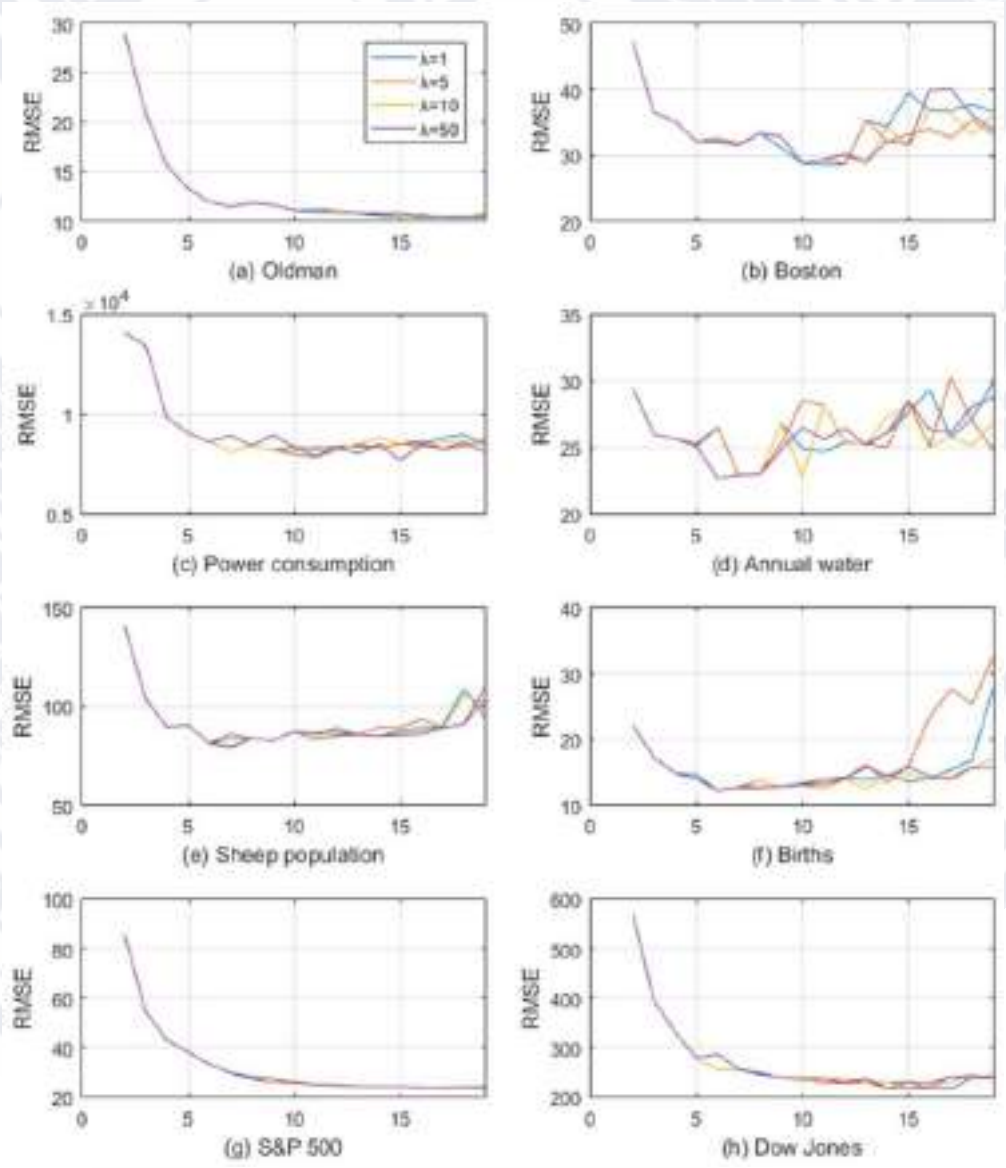

Рис 2.5 – Залежність ефективності НКК від кількості вузлів[32]

2.6 Порівняння з іншими методами прогнозування

Класичні методи прогнозування часових рядів, такі як ARIMA (AutoRegressive Integrated Moving Average) і експоненційне згладжування, традиційно використовуються для моделювання лінійних систем. ARIMA добре справляється з передбаченням стаціонарних часових рядів, де відсутні складні нелінійні залежності. Однак цей метод має обмеження у випадках, коли дані містять нелінійні закономірності або коли система має складну структуру взаємодій між змінними. Для таких ситуацій метод ARIMA може бути недостатньо гнучким, і його прогнози можуть бути неточними. Експоненційне згладжування також ефективне для короткострокових прогнозів, але цей метод менш пристосований до обробки сезонних або сильно мінливих даних.

Ще один класичний метод – SARIMA (Seasonal ARIMA) – є модифікацією ARIMA, який включає сезонні компоненти у свій розрахунок. SARIMA добре працює у випадках, коли дані мають регулярні сезонні коливання, що дозволяє йому бути ефективним для прогнозування таких показників, як погодні умови або економічні індекси з циклічними змінами. Проте в ситуаціях, коли сезонні або інші нелінійні взаємодії між змінними неочевидні або відбуваються з великою невизначеністю, цей метод може не показувати точних результатів.

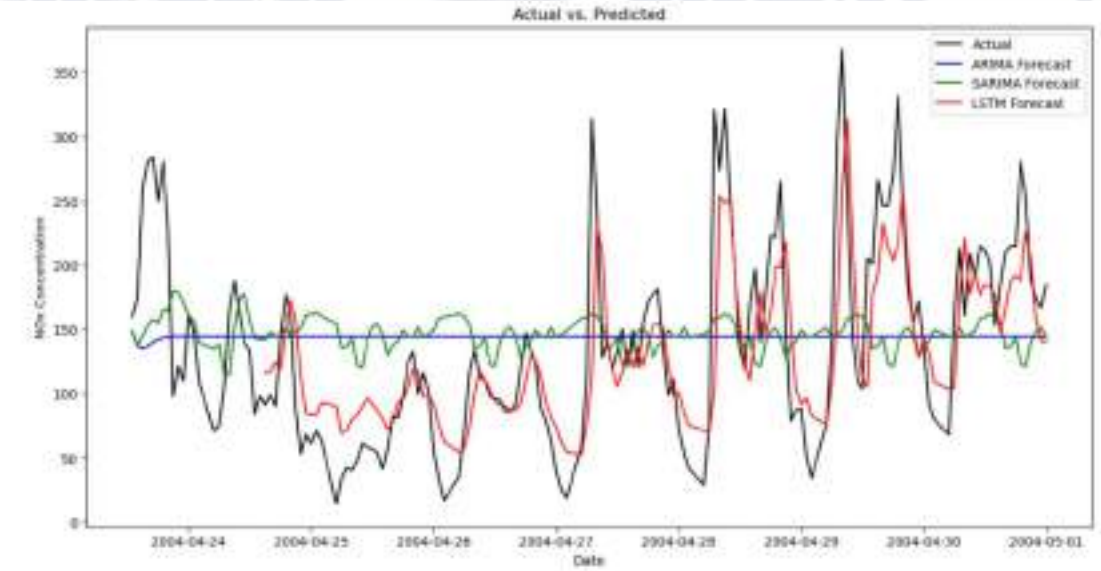

Рис 2.6 – Приклад роботи алгоритмів ARIMA та SARIMA[33]

Ще одним популярним сучасним підходом є методи машинного навчання, такі як Random Forest або XGBoost. Вони здатні будувати моделі прогнозу на основі великої кількості характеристик і взаємодій між ними. Такі методи можуть бути ефективними для прогнозування часових рядів, але вони більше підходять для випадків, де є можливість створити багато незалежних змінних, і можуть бути менш ефективними, коли часові ряди мають високу автокореляцію або коли присутні сильні нелінійні тренди.

Порівняно з нейронними мережами, такими як LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Units), які використовуються для моделювання довгострокових залежностей у часових рядах, НКК також можуть працювати з нелінійними системами, але мають певні переваги. Нейронні мережі є дуже потужним інструментом для роботи з великими обсягами даних і можуть розпізнавати складні патерни. Однак вони потребують великих обчислювальних ресурсів та тривалого процесу навчання, особливо якщо дані мають високу складність або довготривалі тренди. Нейронні мережі також чутливі до якості даних: якщо обсяг даних невеликий або їхня якість низька, це може призвести до поганої точності прогнозів. НКК, навпаки, можуть бути більш адаптивними до обмежених наборів даних, оскільки вони частково ґрунтуються на експертних знаннях, які допомагають моделювати систему навіть за умови відсутності великих обсягів інформації.

У порівнянні з цими підходами, нечіткі когнітивні карти менш вимогливі до обчислювальних ресурсів, ніж нейронні мережі, і можуть бути корисними, коли є необхідність моделювати систему з нечіткими взаємозв’язками між параметрами. Наприклад, у складних соціально-економічних системах, де поведінка учасників або інших компонентів є непередбачуваною або нечітко визначеною, НКК дозволяють створювати прогностичні моделі, які враховують цю невизначеність.

Проте використання НКК також має певні обмеження. Найбільша складність полягає в необхідності точної побудови карти залежностей між концептами та визначенні ваг зв’язків між ними. Це вимагає наявності експертних знань для адекватного визначення взаємодій між компонентами системи. У той

час як ARIMA або нейронні мережі можуть бути повністю автоматизованими та використовувати історичні дані для налаштування моделі, НКК потребують більше часу для побудови, особливо на етапі розробки карти концептів.

Загалом, порівняння НКК з іншими методами показує, що кожен підхід має свої переваги та недоліки, і вибір методу залежить від типу даних, особливостей системи та мети прогнозування. НКК особливо корисні в тих випадках, коли є необхідність моделювати нелінійні, нечіткі залежності між параметрами або коли дані містять невизначеність, яку важко описати математично точними моделями. Водночас для систем із простими лінійними залежностями або коли є великі обсяги даних, нейронні мережі або класичні методи можуть бути ефективнішими в короткостроковій перспективі.

2.7 Стек технологій та інструменти для реалізації моделі

В даній роботі використаний технологічний стек на основі мови програмування Python та низки популярних бібліотек, які забезпечують ефективну обробку даних, візуалізацію, реалізацію нечітких логічних операцій та оптимізаційні алгоритми. Розглянемо детально, як кожна з обраних технологій використовується в контексті дослідження.

Python був обраний як основна мова програмування завдяки своїй популярності у наукових дослідженнях, зокрема у сфері обробки даних та машинного навчання[28]. Python пропонує широкий вибір бібліотек, які дозволяють вирішувати складні математичні задачі, працювати з великими обсягами даних і візуалізувати результати, що є важливими аспектами при роботі з часовими рядами та когнітивними картами.

Бібліотека NumPy відіграє важливу роль у цьому дослідженні, оскільки вона надає функціональні можливості для ефективної роботи з багатовимірними масивами і матрицями. У процесі прогнозування часових рядів, обробка даних часто потребує виконання операцій з масивами, таких як лінійна алгебра, статистичні операції та швидка обробка великих наборів числових даних. NumPy

дозволяє зручно маніпулювати масивами і виконувати такі операції з високою продуктивністю. Крім того, його інтеграція з іншими бібліотеками робить його основою для багатьох математичних і статистичних обчислень у процесі роботи з даними.

Для візуалізації результатів використовується бібліотека Matplotlib. Ця бібліотека дозволяє створювати високоякісні графіки і діаграми, що є особливо корисним при вивченні часових рядів та взаємозв'язків між змінними. За допомогою Matplotlib можна візуалізувати часові ряди, будувати графіки зміни показників у часі, що допомагає інтерпретувати результати моделювання та виявляти патерни у даних. Вона також дозволяє створювати комплексні графіки взаємодії концептів у когнітивних картах, що допомагає візуалізувати структуру і вплив кожного концепту в моделі.

Ключовим елементом у побудові нечітких когнітивних карт є бібліотека skfuzzy, яка надає інструменти для роботи з нечіткою логікою. Використовуючи цю бібліотеку, можна реалізувати нечіткі множини і правила, що лежать в основі когнітивних карт. Skfuzzy дозволяє будувати моделі, які враховують нечіткі залежності між змінними, що є критично важливим для досліджень, де відсутні точні або лінійні взаємозв’язки. За допомогою цієї бібліотеки можна моделювати систему, враховуючи нечіткість у взаємодіях між концептами, і визначати, як зміни одних параметрів впливають на інші у нечіткому середовищі.

Бібліотека CVXPY використовується для вирішення задач оптимізації. У контексті побудови нечітких когнітивних карт та прогнозування часових рядів часто виникає потреба в оптимізації параметрів системи, таких як ваги зв'язків між концептами або інші коефіцієнти, які визначають динаміку взаємодій. CVXPY дозволяє вирішувати задачі лінійної і нелінійної оптимізації, що є важливим для налаштування моделі з урахуванням мінімізації похибок прогнозування. Оптимізація може застосовуватися як для налаштування внутрішніх параметрів системи, так і для знаходження найкращих рішень на основі обмежень, які накладаються на модель.

Sklearn (Scikit-learn) є однією з найбільш популярних бібліотек для машинного навчання в Python. В рамках цього дослідження sklearn використовується для виконання ряду завдань, пов'язаних з аналізом даних, зокрема для розділення даних на тренувальні і тестові набори, крос-валідації, а також для застосування методів регресії або класифікації, які можуть бути необхідними для порівняння з іншими методами прогнозування. Sklearn також надає зручні інструменти для оцінки якості прогнозів, такі як метрики точності (наприклад, середньоквадратична помилка), що дозволяє об’єктивно оцінити продуктивність моделі на основі НКК.

Нарешті, бібліотека SciPy використовується для вирішення різноманітних математичних та статистичних задач, які можуть виникнути в процесі роботи з даними. SciPy є доповненням до NumPy і розширює його функціональні можливості, пропонуючи додаткові інструменти для оптимізації, інтеграції, числових методів і обробки сигналів. Це особливо корисно для роботи з часовими рядами, де часто виникає потреба в обчисленні складних математичних функцій або для вирішення специфічних проблем у контексті моделювання часових рядів.

2.8 Вибір даних для дослідження

У рамках роботи можливі різні предметні області, де прогнозування часових рядів є важливою задачею. Серед них можуть бути економічні, соціальні, технічні та природні процеси, які мають часову динаміку. Одним із варіантів є використання даних з фінансової сфери, наприклад, історичних курсів валют. Однак, прогнозування також може охоплювати інші галузі, такі як аналіз кліматичних змін, прогнозування споживання енергії, аналіз поведінки користувачів в Інтернеті, прогнозування попиту на продукцію або соціально-економічні показники.

Розглянемо кілька можливих прикладів предметних областей та даних, які можуть бути використані в дослідженні:

- фінансова сфера (курси валют, ціни на акції, економічні показники). Для фінансового прогнозування можна використовувати історичні дані про курси валют, ціни на акції, економічні. Ці дані можуть бути щоденними або щомісячними, залежно від специфіки задачі. У фінансових часових рядах часто спостерігаються коливання, сезонні ефекти та тренди, що робить їх зручними для моделювання за допомогою методів, які враховують взаємодію змінних, як це роблять нечіткі когнітивні карти. Прогнозування курсів валют, наприклад, може допомогти банкам і компаніям передбачати майбутні фінансові ризики.
- енергетика (споживання електроенергії, ціни на нафту або газ). В енергетичній сфері можна досліджувати історичні дані про споживання електроенергії на різних ринках або зміну цін на нафту й газ. Такі дані мають важливе значення для планування виробництва та розподілу енергетичних ресурсів. Наприклад, прогнозування майбутнього споживання електроенергії може базуватися на часових рядах, які відображають сезонність та змінюваність попиту. Ці часові ряди також можуть бути пов'язані з іншими змінними, такими як погодні умови або економічна активність, що можна врахувати при побудові моделей на основі НКК.
- кліматичні дані (температура, опади, атмосферний тиск). Прогнозування кліматичних умов, таких як температура повітря або кількість опадів, може бути критично важливим для різних галузей, включаючи сільське господарство, будівництво, транспорт і туризм. Історичні кліматичні дані, які можна збирати з метеорологічних станцій або супутникових спостережень, містять тренди, сезонні коливання та випадкові коливання. Використання нечітких когнітивних карт дозволяє моделювати складні залежності між кліматичними показниками і отримувати прогнози з урахуванням взаємозв’язків між ними.
- технічні процеси (прогнозування відмов технічних систем, аналіз IoT даних). У технічних системах важливим аспектом є прогнозування відмов

або проблем у роботі обладнання. Дані про технічний стан систем можуть включати інформацію про кількість відмов, виробничі цикли або параметри роботи обладнання в часі. Наприклад, в індустрії IoT (Інтернету речей) накопичуються дані з сенсорів, які відображають роботу технічних пристроїв. Прогнозування можливих відмов або проблем в експлуатації систем дозволяє запобігти поломкам і підвищити ефективність управління технічними ресурсами.

РОЗДІЛ 3. КОМП’ЮТЕРНИЙ ЕКСПЕРИМЕНТ

У цьому розділі буде розглянуто розроблену комп’ютерну програму для відновлення фазового простору і виділення нечіткої когнітивної карти з часового ряду для його прогнозування, а також результати її роботи. Буде розглянуто 3 часові ряди з різних предметних областей, таких як: енергетика(споживання електроенергії), кліматичні дані(атмосферний тиск) та втрати російської армії під час вторгнення в Україну(кількість знищених бойових броньованих машин).

Алгоритм навчається на перших 80 відсотках даних і прогнозує решту 20. На графіках буде зображено порівняння прогнозованого та реального значення тестових даних, а також порівняння прогнозованого значення з усім часовим рядом.

3.1 Атмосферний тиск

Weather Dataset - це набір даних часових рядів з інформацією про погоду в певному місці за годину. Він записує температуру, температуру точки роси, відносну вологість, швидкість вітру, видимість, тиск і умови[35]. Розглядатимемо з цього датасету перші 2000 записів ряду атмосферного тиску(Press_kPa).

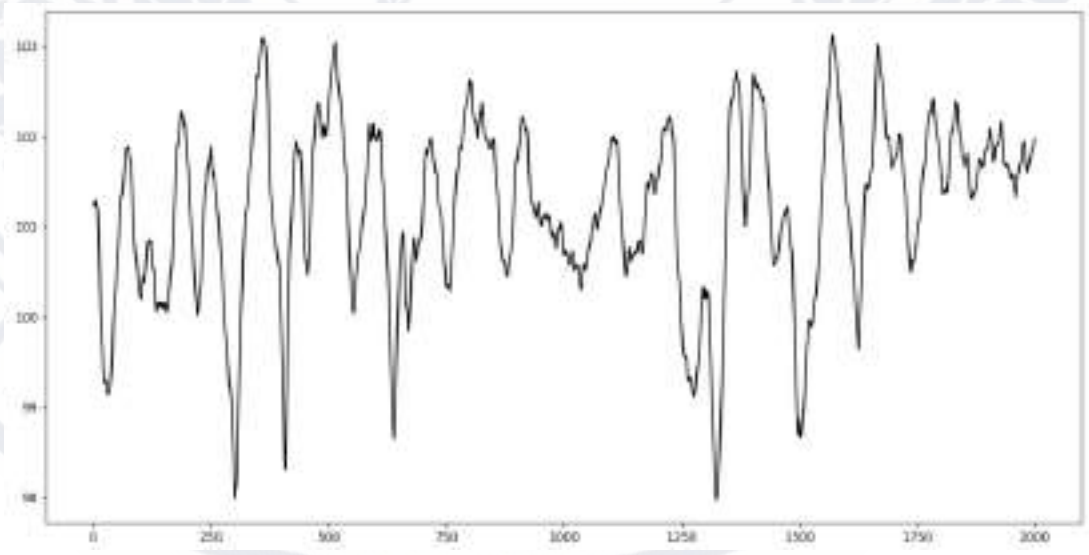

Рис 3.1 – Часовий ряд атмосферного тиску

Оптимальна часова затримка: за графіком функції автокореляції (рис. 3.2) обираємо значення τ, де значення графіка найменше. τ = 8.

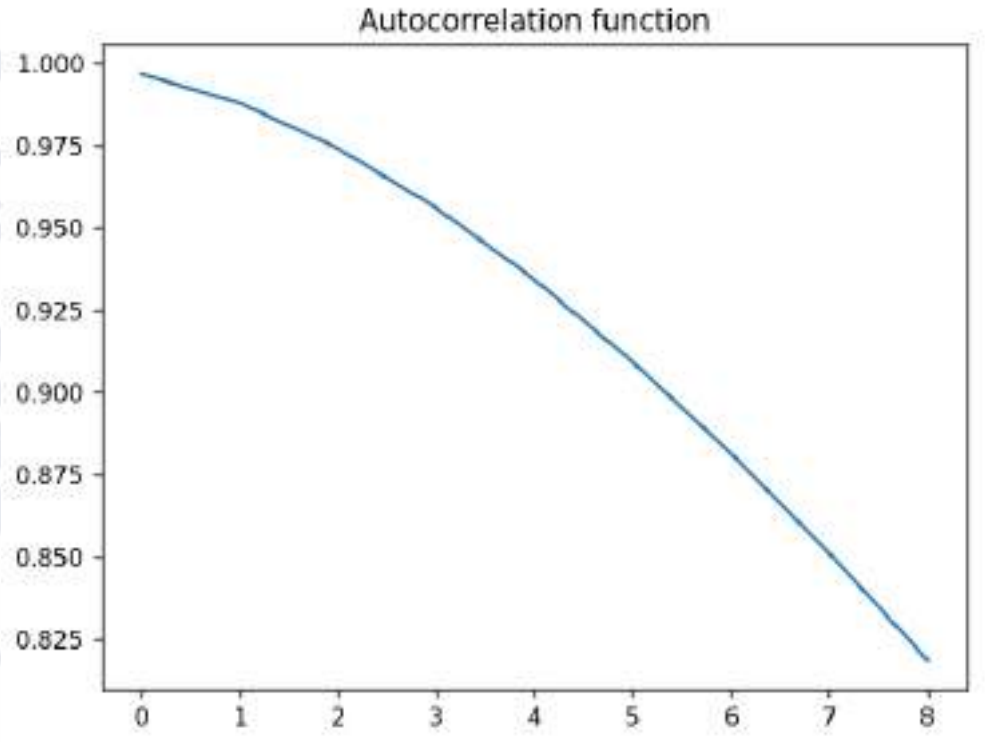

Рис 3.2 – Графік функції автокореляції для Press_kPa

Оптимальна кількість вимірів: за графіком пропорції найближчих хибних сусідів від кількості вимірів (рис. 3.3), обираємо d = 7.

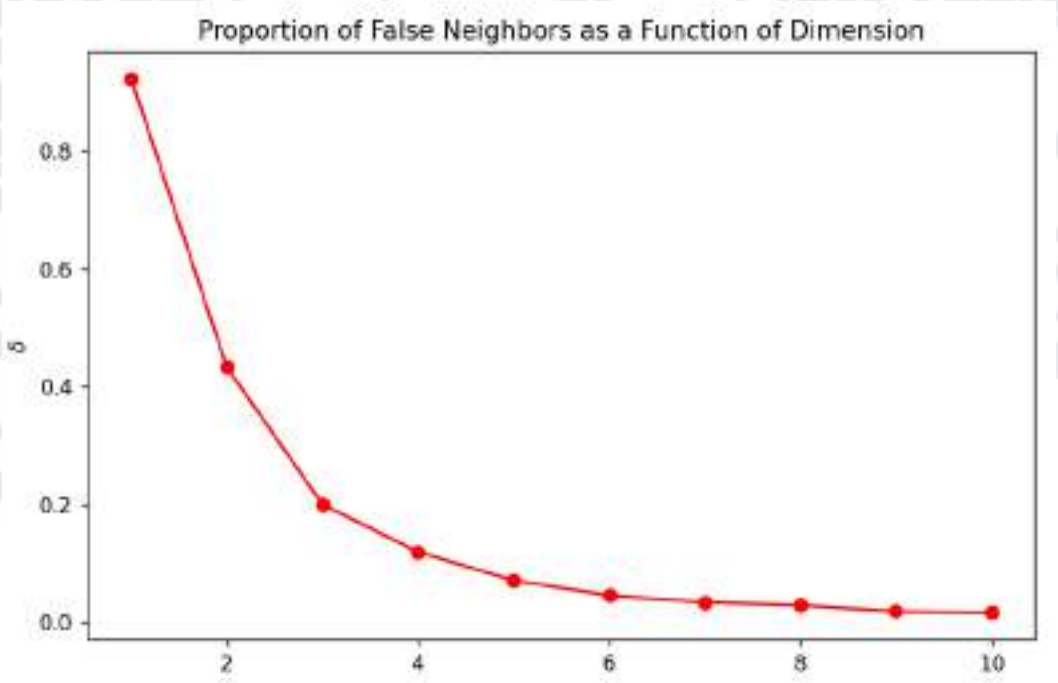

Рис 3.3 – Залежність False Nearest Neighbors від кількості вимірів

Показник Ляпунова для даного ряду дорівнює 0.077, що свідчить про його хаотичну поведінку.

На основі отриманих даних можемо побудувати відновлений фазовий портрет.

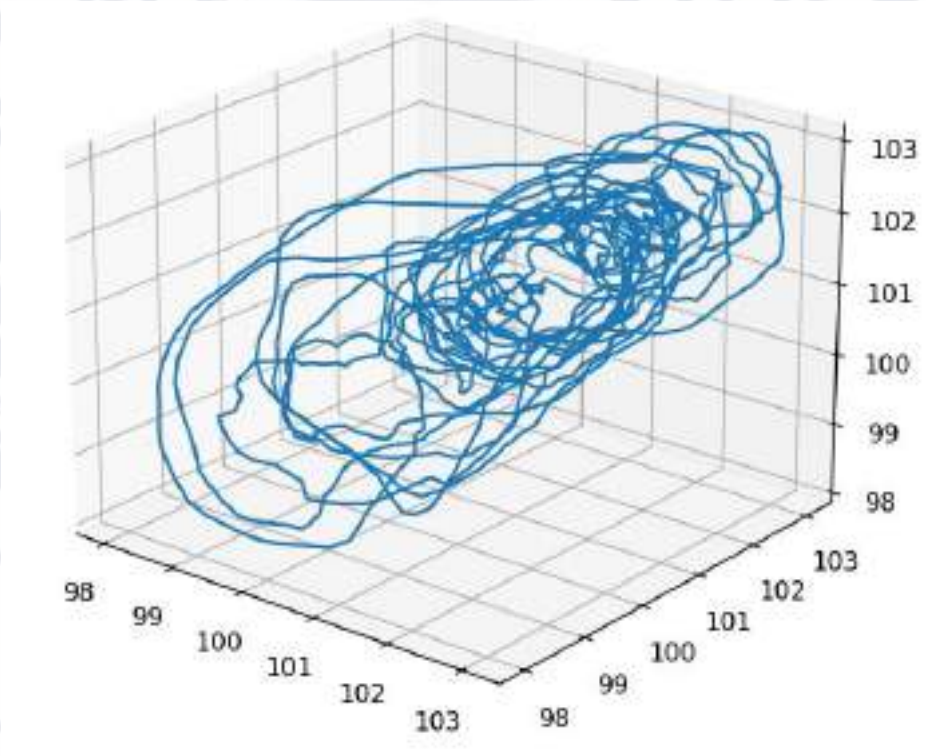

Рис 3.4 – Відновлений фазовий портрет для Press_kPa

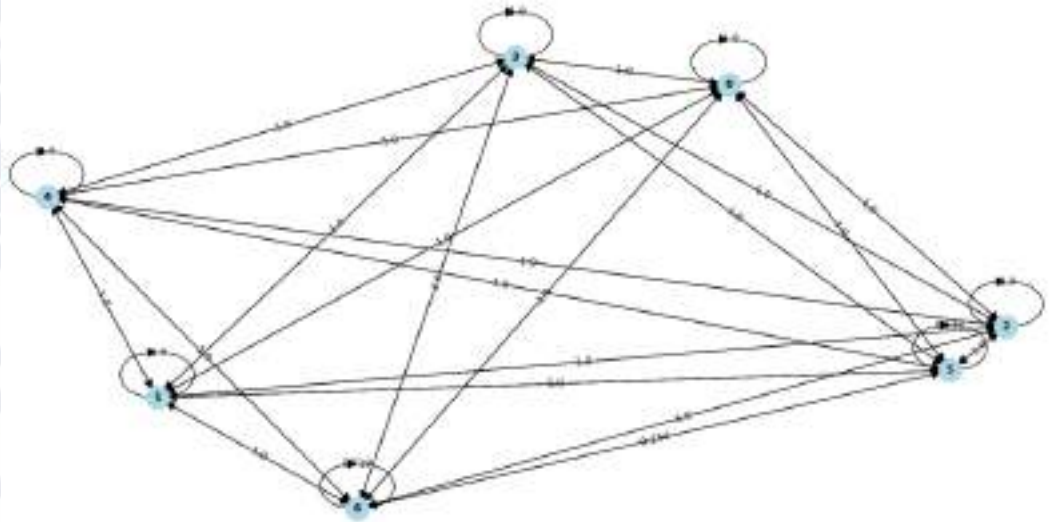

Рис 3.5 – Структура нечіткої когнітивної карти для Press_kPa

Маємо наступний результат прогнозування:

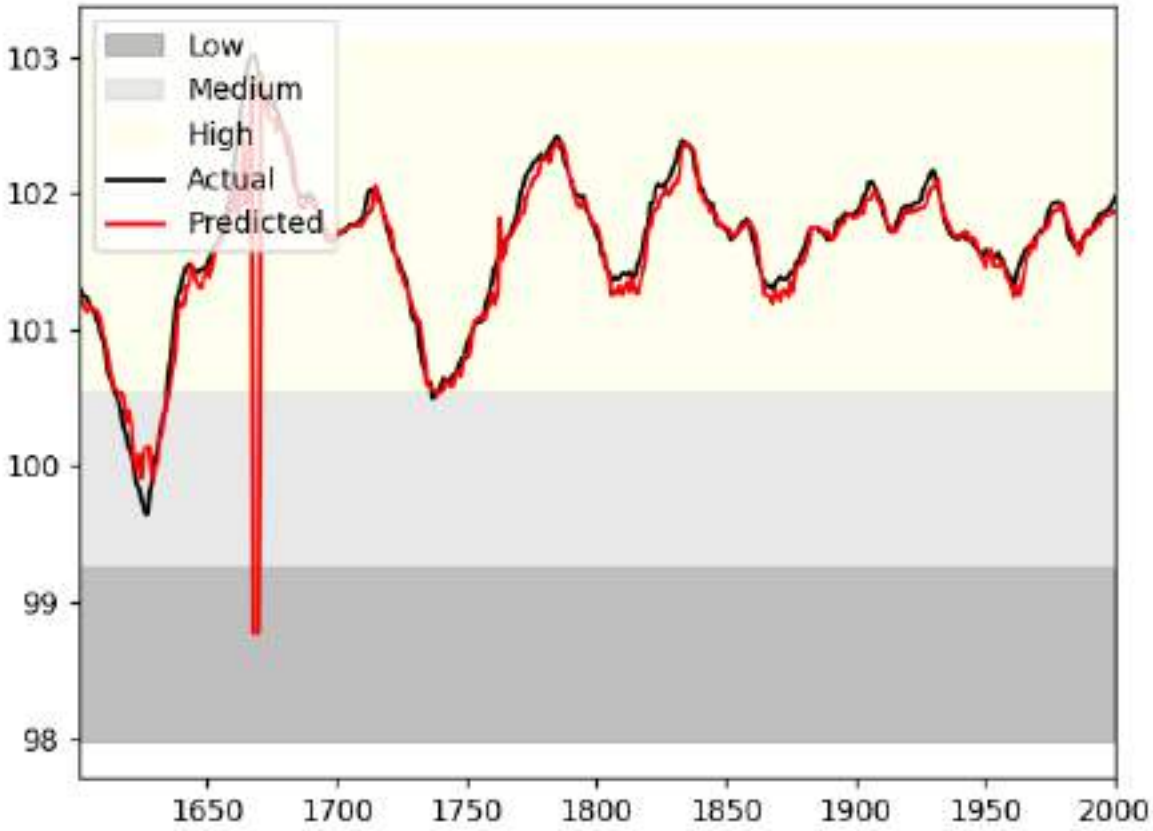

Рис 3.6 – Прогнозовані та реальні дані останніх 20 відсотків часового ряду

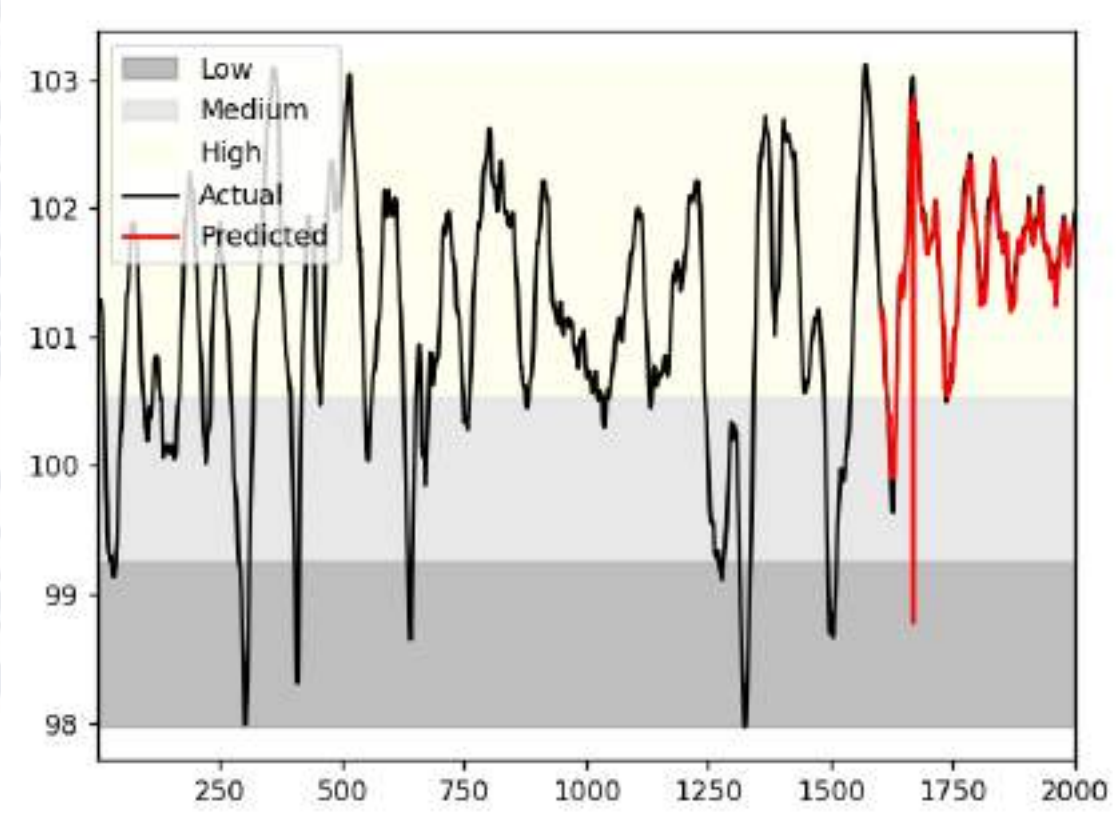

Рис 3.7 – Прогнозовані дані у порівнянні з усім часовим рядом

Таблиця 3.1

Кількість концептів	RMSE
5	0.5654
6	0.6025
7	**0.5266**
8	0.7094

У таблиці 3.1 показано залежність середньоквадратичного відхилення прогнозованих та реальних даних від кількості концептів НКК. Бачимо, що найменше значення похибки отримується при кількості концептів 7. Тож, можна зробити висновок про доцільність інтеграції теорії хаосу та нечітких когнітивних карт.

3.2 Споживання електроенергії

Energy Consumption Time Series Dataset – це набір даних, що містить витрату електроенергії споживачами з часовими точками близько 10-15 хв, які записуються за допомогою пристрою IoT.

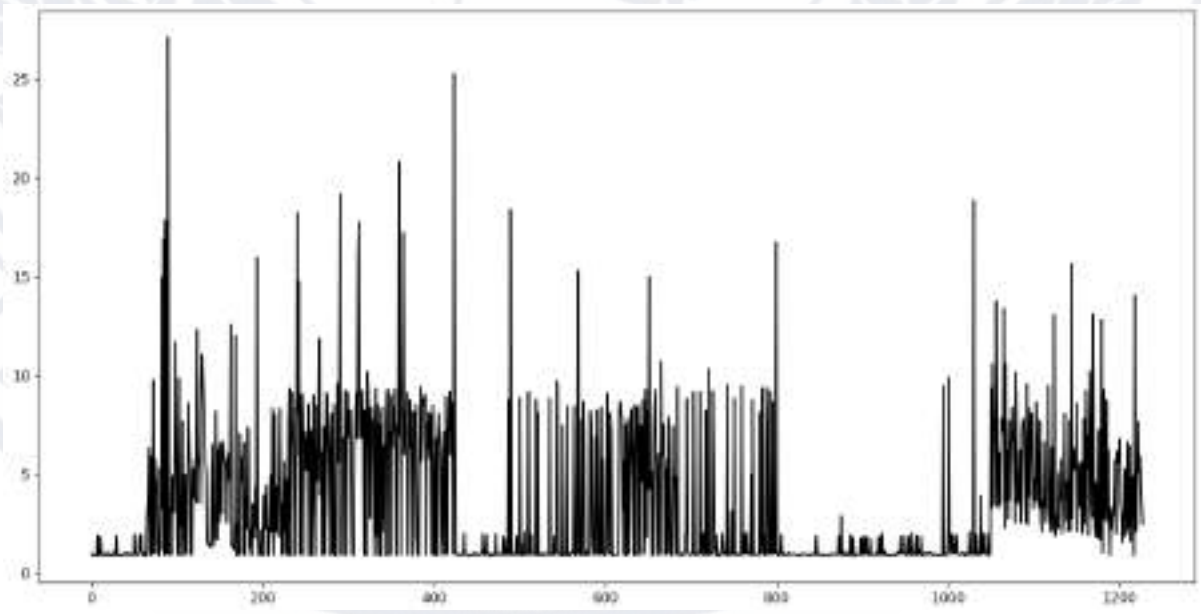

Рис 3.8 – Часовий ряд споживання електроенергії

З графіків автокореляції (рис. 3.9) та залежності False Nearest Neighbors від кількості вимірів (рис. 3.10) обираємо значення τ = 7, d = 8. Показник Ляпунова – 0.122.

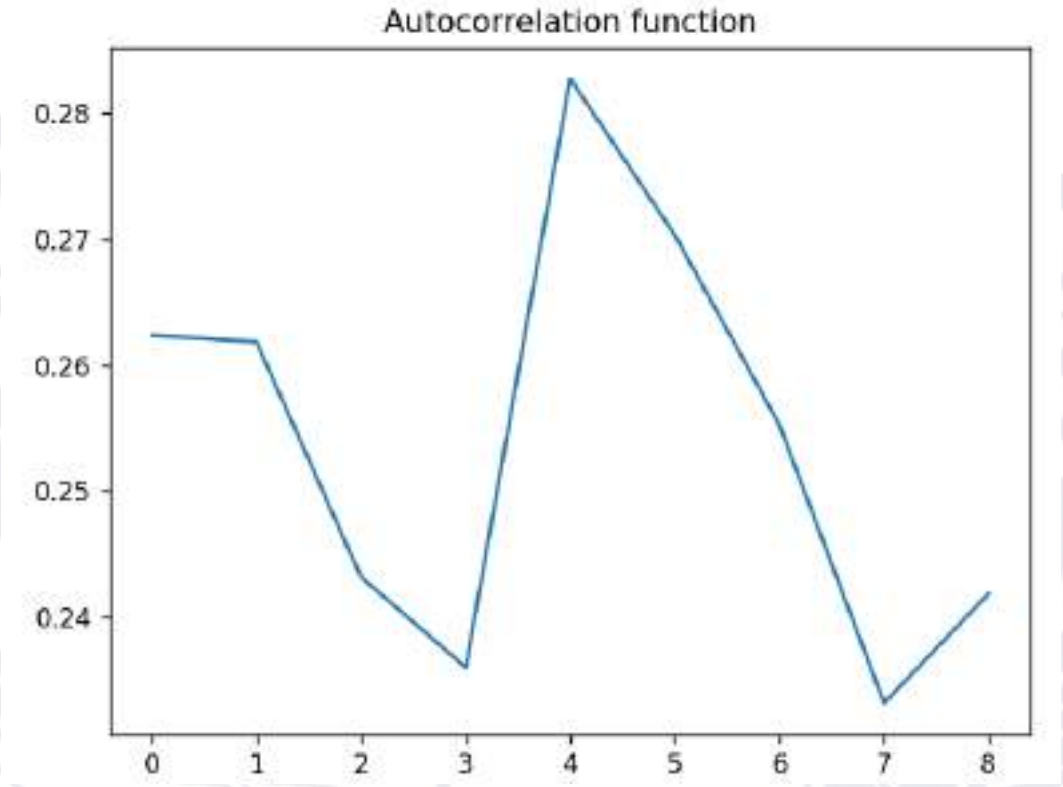

Рис 3.9 – Графік функції автокореляції

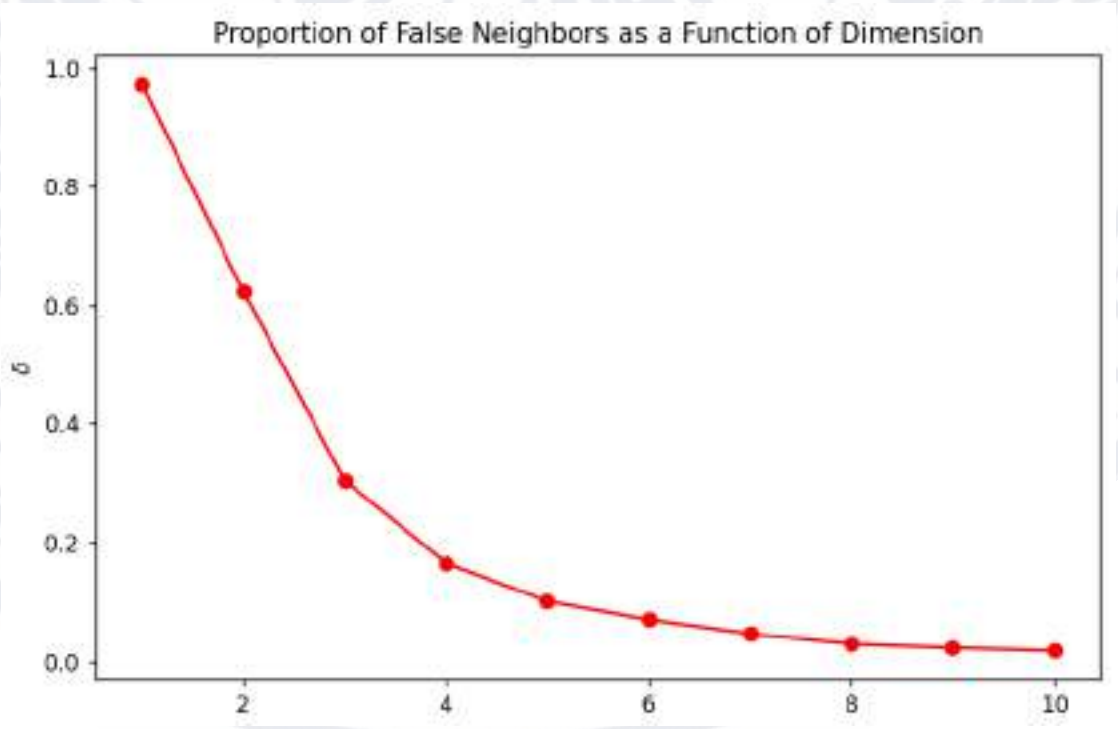

Рис 3.10 – Залежність False Nearest Neighbors від кількості вимірів

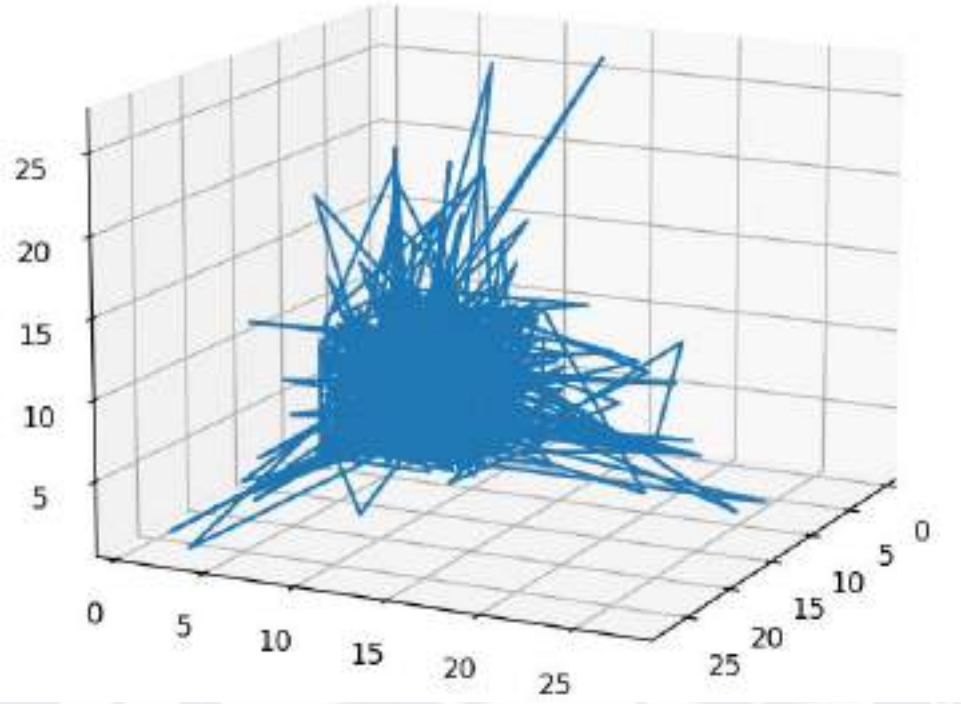

Рис 3.11 – Відновлений фазовий портрет

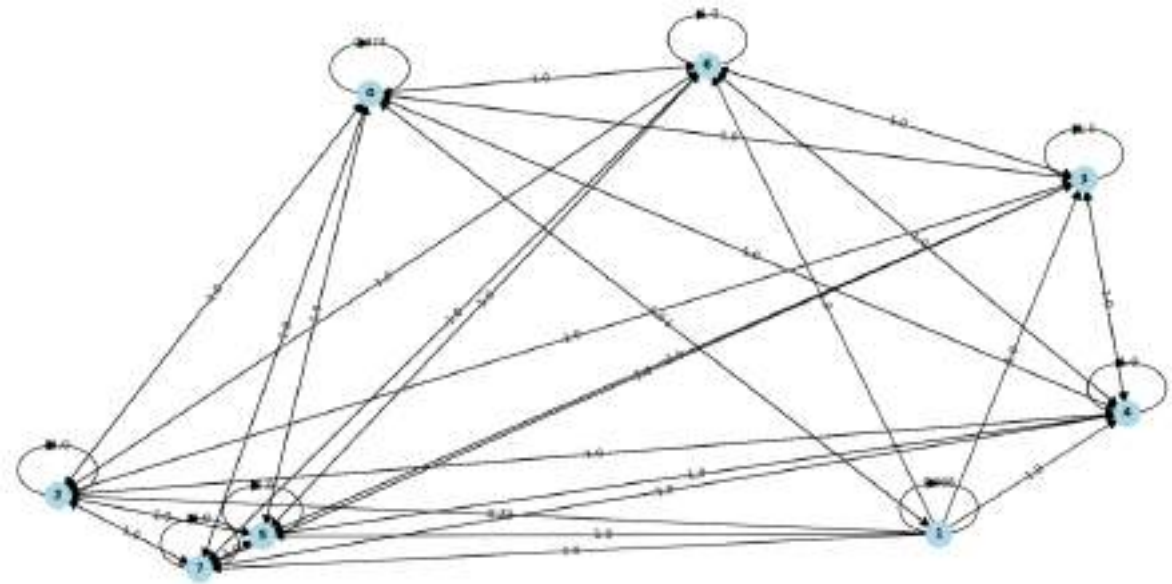

Рис 3.12 – Структура нечіткої когнітивної карти

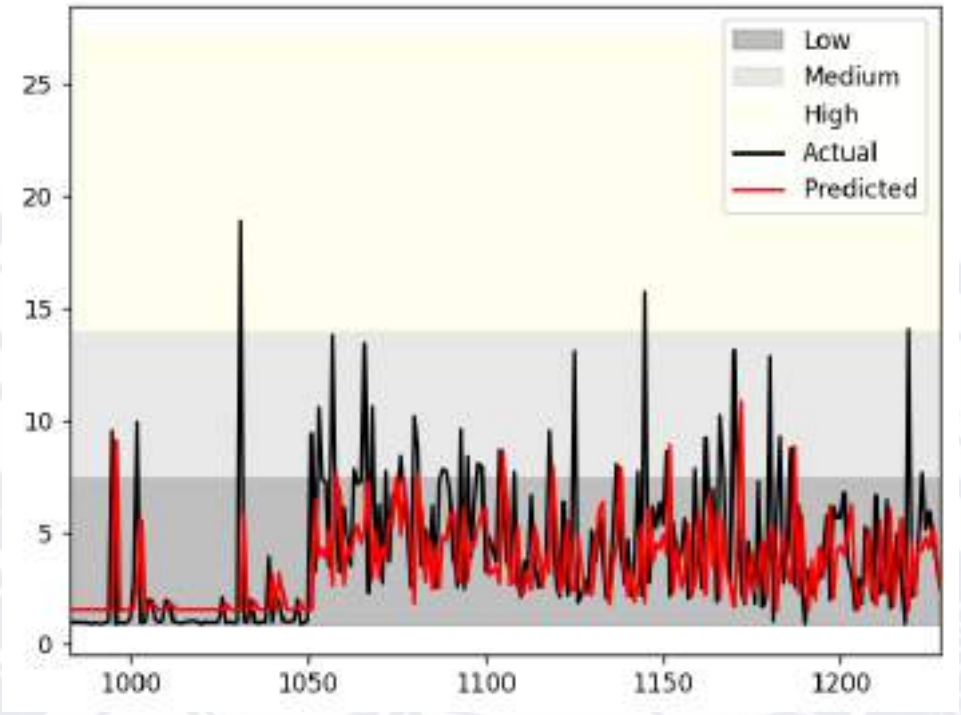

Рис 3.13 – Прогнозовані та реальні дані останніх 20 відсотків часового ряду

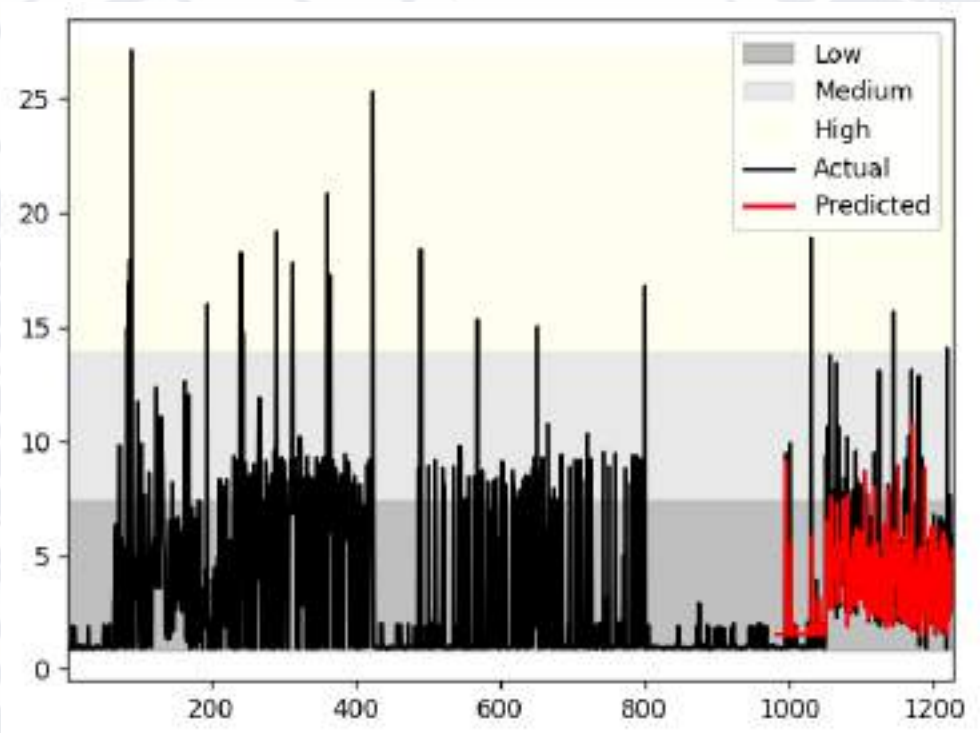

Рис 3.14 – Прогнозовані дані у порівнянні з усім часовим рядом

Таблиця 3.2

Кількість концептів	RMSE
6	0.7713
7	0.6583
8	**0.4379**
9	0.5768

Найменше значення середньоквадратичного відхилення отримуємо при значенні кількості концептів 8.

3.3 Втрати ворога

2022 Russia Ukraine War – це набір даних, що містить втрати ворогом техніки за час російсько-української війни.

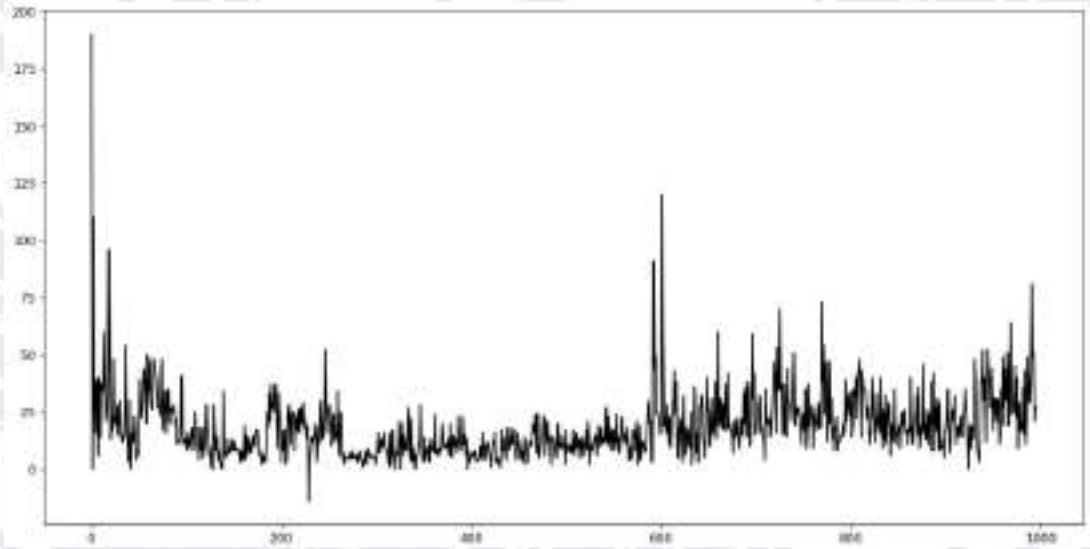

Рис 3.15 – Часовий ряд втрат ворога за днями

З графіків 3.16 та 3.17 отримуємо значення $\tau = 8$, $d = 4$. Показник Ляпунова – 0.242.

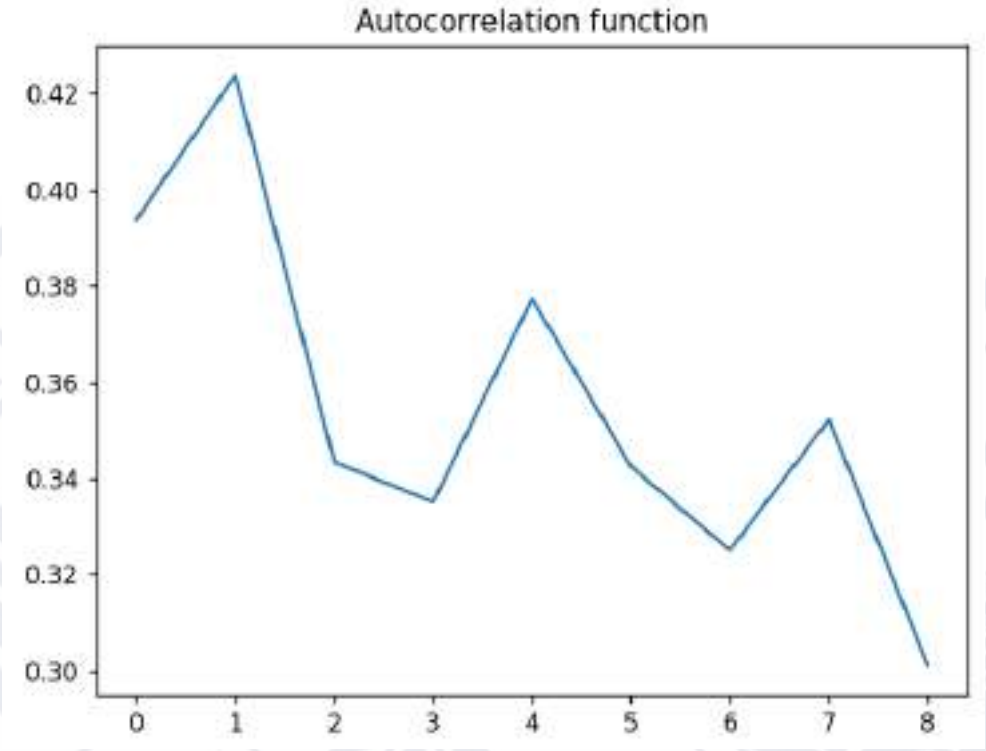

Рис 3.16 – Графік функції автокореляції

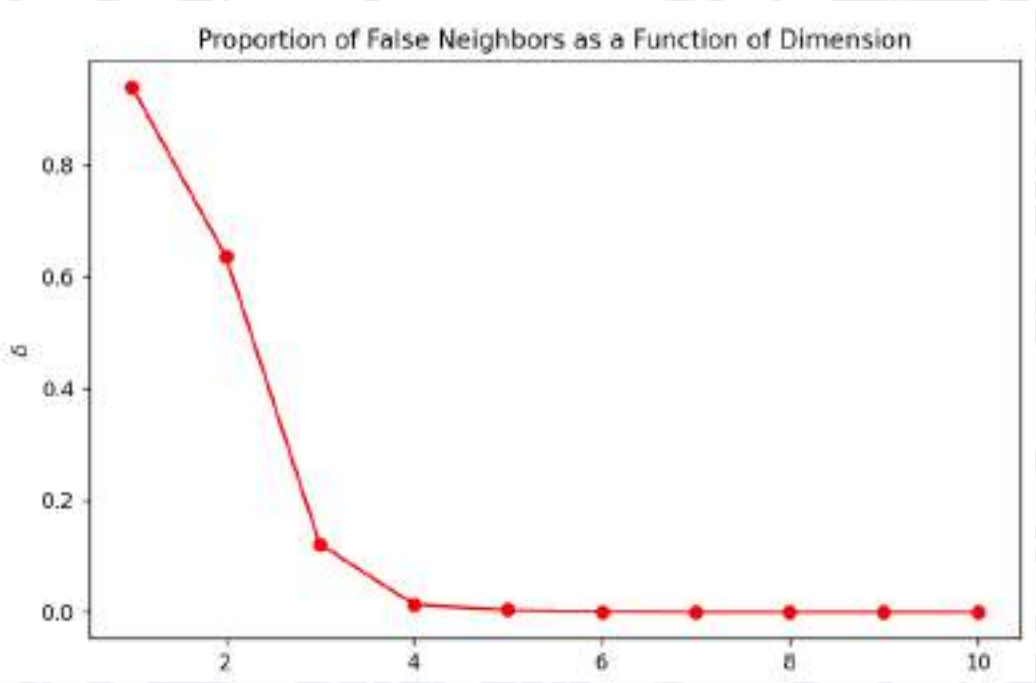

Рис 3.17 – Залежність False Nearest Neighbors від кількості вимірів

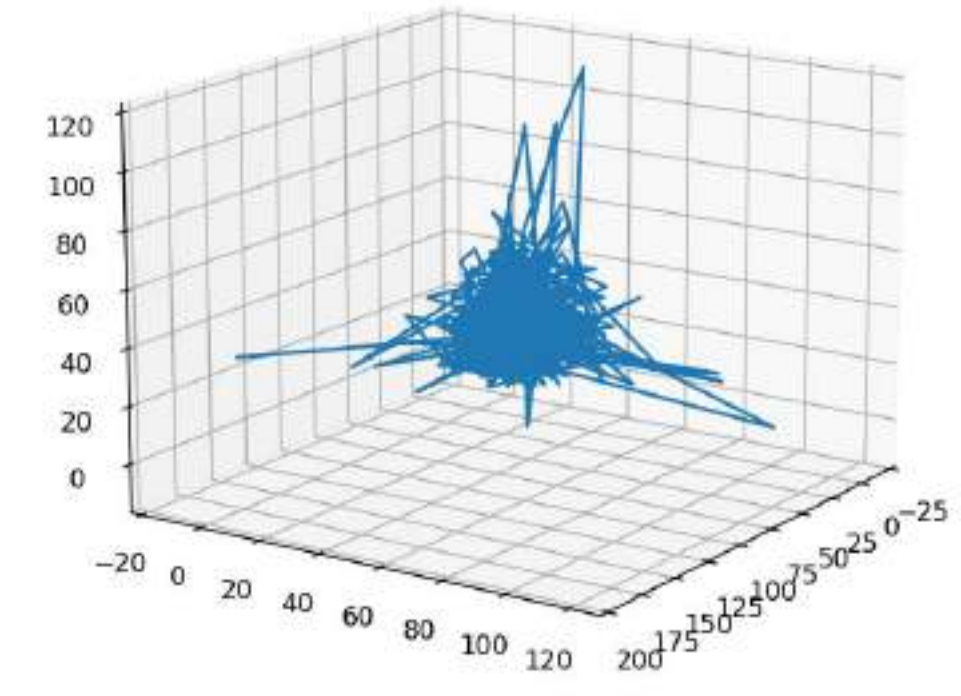

Рис 3.18 – Відновлений фазовий портрет

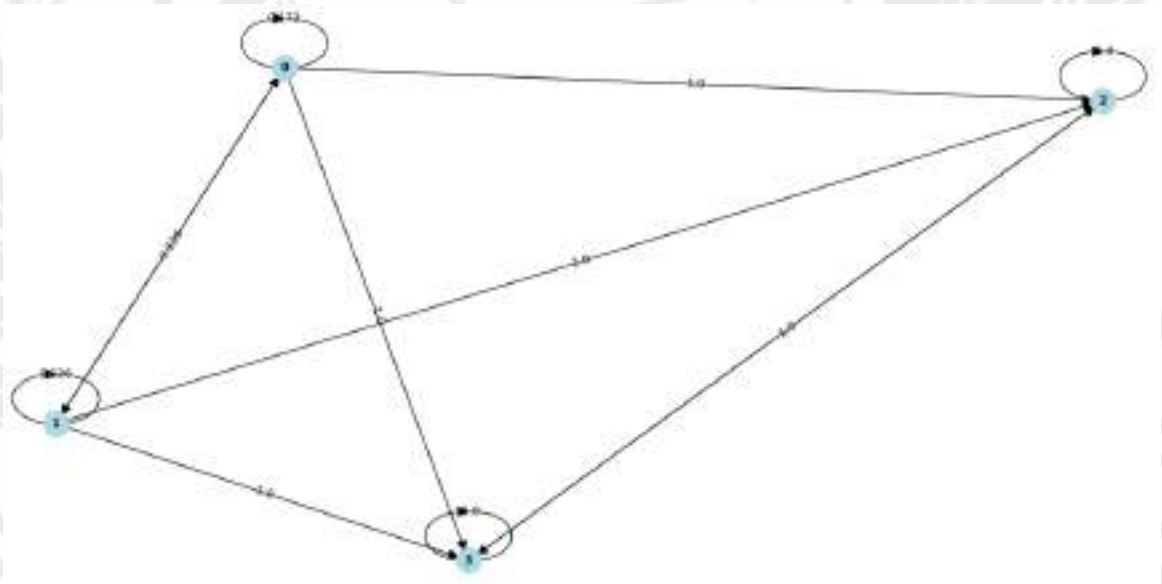

Рис 3.19 – Структура нечіткої когнітивної карти

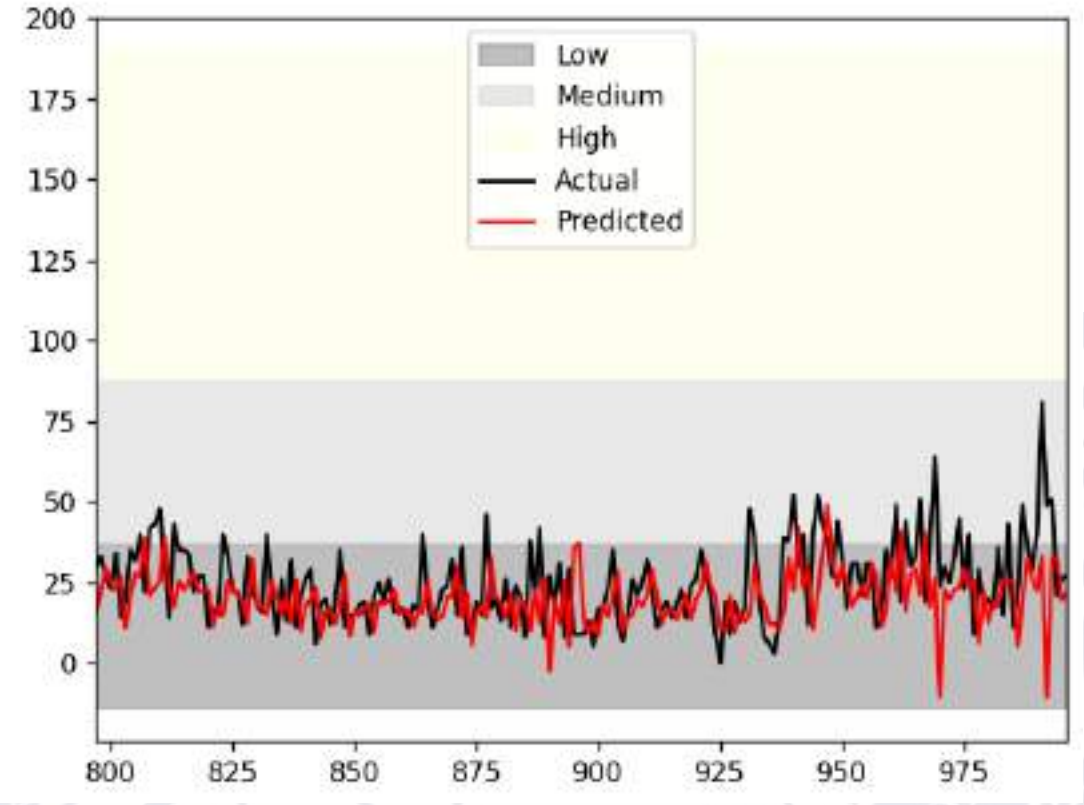

Рис 3.20 – Прогнозовані та реальні дані останніх 20 відсотків часового ряду

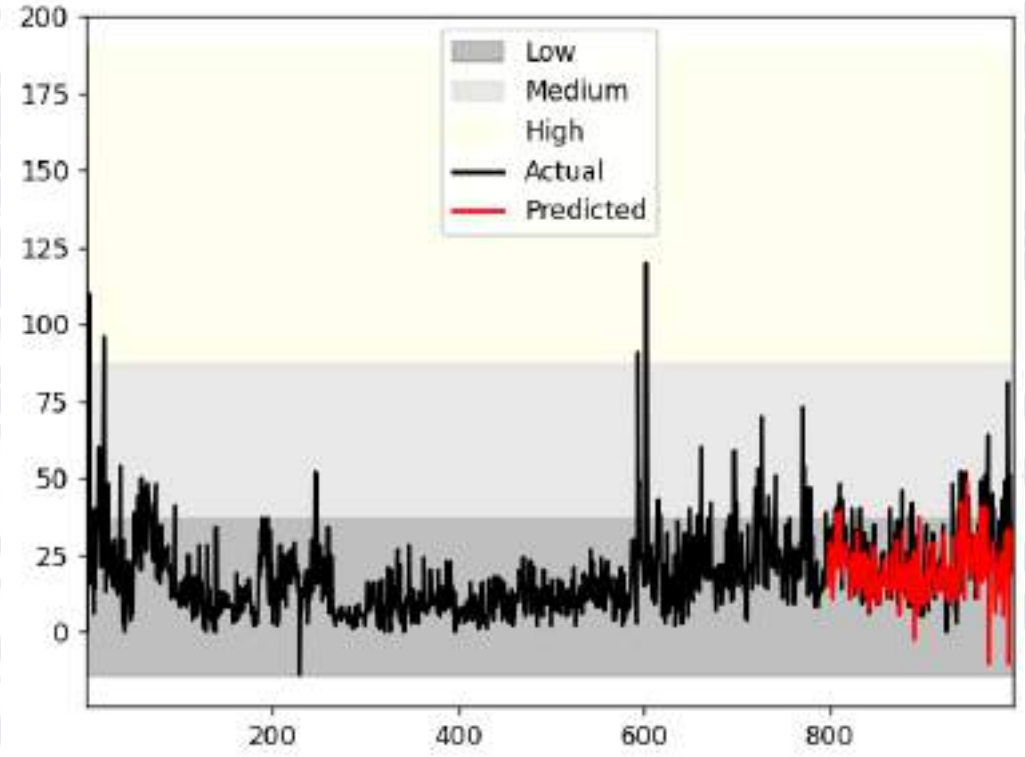

Рис 3.21 – Прогнозовані дані у порівнянні з усім часовим рядом

Таблиця 3.3

Кількість концептів	RMSE
3	15.0102
4	**14.8494**
5	14.8538
6	14.9239

Найменше значення середньоквадратичного відхилення отримуємо при значенні кількості концептів 4.

3.4 Порівняння з іншими методами прогнозування

Наведемо порівняння представленого методу з найбільш популярними методами прогнозування часових рядів: moving average(рухоме середнє) та ARIMA(авторегресійне рухоме середнє). Порівнюватимемо за критерієм RMSE між прогнозованими та реальними даними.

Таблиця 3.4

Метод	Атмосферний тиск	Споживання електроенергії	Втрати ворога
Moving average	0.7447	4.4899	15.2980
ARIMA	0.5906	3.5687	14.4804
НКК	0.5266	0.4379	14.8494

Результати таблиці 3.4 показують, що для певних наборів даних використання нечітких когнітивних карт дає переваги порівняно з класичними методами.

ВИСНОВКИ

Було продемонстровано можливості нечітких когнітивних карт (НКК) як ефективного інструменту для прогнозування часових рядів. Дана тема має великий потенціал у вирішенні складних задач прогнозування в різних галузях.

Було розглянуто алгоритм виділення нечіткої когнітивної карти з часового ряду на основі історичних спостережень з усіма суміжними етапами: формування нечіткої моделі часового ряду, створення нечіткої когнітивної карти на основі цієї моделі, її прогнозування та формування реальних прогнозованих даних з нечіткої моделі .

Цей підхід поєднує в собі переваги нечіткої логіки, яка дозволяє ефективно моделювати нечіткі та неоднорідні взаємозв'язки в даних, з когнітивними картами, які дозволяють репрезентувати знання експертів та враховувати їхні інтуїтивні уявлення про систему. Це дозволяє створювати більш точні та адаптивні моделі для прогнозування майбутніх значень часових рядів.

Однією з основних переваг підходу є його гнучкість і універсальність. Нечіткі когнітивні карти можуть бути успішно застосовані до різноманітних типів даних та задач прогнозування, включаючи економічні, фінансові, кліматичні та інші. Це робить їх потужним інструментом для рішення реальних проблем у різних галузях.

Варто зазначити, що для досягнення повного потенціалу цього підходу потрібно ще провести значний обсяг досліджень. Це включає в себе подальший розвиток методів навчання та оптимізації моделей, а також розробку ефективних стратегій валідації та перевірки точності прогнозів. Крім того, важливо звернути увагу на можливість використання цих моделей у реальних умовах та їхню адаптацію до конкретних ситуацій.

В цілому, прогнозування часових рядів за допомогою нечітких когнітивних карт є перспективним напрямком досліджень, який може допомогти вирішувати складні проблеми прогнозування в різних галузях.

Успішне впровадження та подальший розвиток цього підходу можуть мати значний вплив на практичне застосування прогнозування та аналізу даних у майбутньому. Зокрема, це може сприяти покращенню точності прогнозів у різних галузях, таких як фінанси, економіка, медицина, технології та інші. Такий підхід може допомогти вирішувати складні проблеми прогнозування, пов'язані з нестаціонарністю, шумом у даних, великою кількістю змінних та іншими складнощами, що зустрічаються в реальних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. B. Kosko. Fuzzy cognitive maps. International journal of man-machine studies, 1986, 24.1: 65-75 pp.
2. R. Axelrod. Structure of decision: The cognitive maps of political elites. Princeton university press, 2015. 422 p.
3. Y. Li, Y. Shao. Fuzzy cognitive maps based on D-number theory. IEEE Access, 2022, 10: 72702-72716.
4. G. Papakostas, D. Koulouriotis, Classifying patterns using fuzzy cognitive maps, in: Fuzzy cognitive maps, Springer, 2010. 291–306 pp.
5. G. Felix, G. N´apoles, R. Falcon, W. Froelich, K. Vanhoof, R. Bello, A review on methods and software for fuzzy cognitive maps, Artificial intelligence review 52, 2019. 1707–1737 pp.
6. A. K. Tsadiras, Comparing the inference capabilities of binary, trivalent and sigmoid fuzzy cognitive maps. Information Sciences, 2008, 178.20: 3880-3894 pp.
7. S. Bueno, J. L. Salmeron. Benchmarking main activation functions in fuzzy cognitive maps. Expert systems with Applications, 2009, 36.3: 5221-5229 pp.
8. E. I. Papageorgiou. Learning algorithms for fuzzy cognitive maps. In: EUSFLAT Conf. 2001. 83-88 pp.
9. J. A. Dickerson, B. Kosko. Virtual worlds as fuzzy cognitive maps. Presence: Teleoperators & Virtual Environments, 1994, 3.2: 173-189 pp.
10. J. L. Salmeron, T. Mansouri, M. R. S. Moghadam, A. Mardani. . Learning fuzzy cognitive maps with modified asexual reproduction optimisation algorithm. Knowledge-Based Systems, 2019, 163: 723-735 pp.
11. A. V. Huerga. A balanced differential learning algorithm in fuzzy cognitive maps. In: Proceedings of the 16th international workshop on qualitative reasoning. 2002. 7 p.

12. E. Papageorgiou, C. Stylios, P. Groumpos, Active hebbian learning algorithm to train fuzzy cognitive maps. . International journal of approximate reasoning, 2004, 37.3: 219-249 pp.
13. E. Papageorgiou, C. Stylios, P. Groumpos. Fuzzy cognitive map learning based on nonlinear Hebbian rule. In: AI 2003: Advances in Artificial Intelligence: 16th Australian Conference on AI, Perth, Australia, December 3-5, 2003. Proceedings 16. Springer Berlin Heidelberg, 2003. 256-268 pp.
14. J. Salmeron, E. Papageorgiou. Fuzzy grey cognitive maps and nonlinear Hebbian learning in process control. Applied intelligence, 2014, 41: 223-234 pp.
15. K. Parsopoulos, E. Papageorgiou, M. Vrahatis. A first study of fuzzy cognitive maps learning using particle swarm optimization. In: The 2003 Congress on Evolutionary Computation, 2003. CEC'03. IEEE, 2003. 1440-1447 pp.
16. W. Stach, L. Kurgan, W. Pedrycz, M. Reformat. Genetic learning of fuzzy cognitive maps. Fuzzy sets and systems, 2005, 153.3: 371-401 pp.
17. N. H. Mateou, M. Moiseos, A. S. Andreou. Multi-objective evolutionary fuzzy cognitive maps for decision support. In: 2005 IEEE Congress on Evolutionary Computation. IEEE, 2005. 824-830 pp.
18. D. Koulouriotis, I. Diakoulakis, D. Emiris. Learning fuzzy cognitive maps using evolution strategies: a novel schema for modeling and simulating high-level behavior. In: Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546). IEEE, 2001. 364-371 pp.
19. C. Lin. An immune algorithm for complex fuzzy cognitive map partitioning. In: Proceedings of the first ACM/SIGEVO summit on genetic and evolutionary computation. 2009. 315-320 pp.
20. E. I. Papageorgiou, P. P. Groumpos. A new hybrid method using evolutionary algorithms to train fuzzy cognitive maps. Applied Soft Computing, 2005, 5.4: 409-431 pp.
21. A. Rotshtein, L. Pustylnik, Y. Giat. Fuzzy Logic and Chaos Theory in Time Series Forecasting. International Journal of Intelligent Systems, 2016.
22. The phase portraits

URL: https://www.researchgate.net/figure/The-phase-portraits-of-Eq-33-a-3D-phase-portrait-with-A-0-3-5_fig2_383252393

23. E. I. Papageorgiou, P. P. Groumpos. A new hybrid method using evolutionary algorithms to train fuzzy cognitive maps. Applied Soft Computing, 2005, 5.4: 409-431 pp.
24. Stach W, Kurgan LA, Pedrycz W. Numerical and linguistic prediction of time series with the use of fuzzy cognitive maps. IEEE Transactions on Fuzzy Systems, 2008, 16(1):61–72
25. Triangular fuzzy set
URL: https://www.researchgate.net/figure/Triangular-fuzzy-set_fig1_220467426
26. H. Kato. Takens-type reconstruction theorems of one-sided dynamical systems. 2022.
27. S. Wallot, D. Monster. Calculation of Average Mutual Information and False-Nearest Neighbors for the Estimation of Embedding Parameters of Multidimensional Time Series in Matlab. Frontiers in Psychology 9, 2018.
28. Autocorrelation functions
URL: https://www.researchgate.net/figure/a-Autocorrelation-functions-for-some-time-series-analyzed-in-this-work-The-temporal_fig2_244135015
29. M. B. Kennel, R. Brown, H. D. I. Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction. 1992.
30. False nearest neighbor (FNN) versus embedding dimension (on the left) and autocovariance function versus delay (on the right) for a Lorenz hyperchaotic attractor filtered by the 8-DOF system.
URL: https://www.researchgate.net/figure/False-nearest-neighbor-FNN-versus-embedding-dimension-on-the-left-and-autocovariance_fig6_228942730
31. Boyd S, Vandenberghe L. Convex optimization. Cambridge: Cambridge University Press. 2004.
32. The performance of the fuzzy cognitive map model with different parameters
URL: https://www.researchgate.net/figure/The-performance-of-the-fuzzy-cognitive-map-FCM-model-with-different-parameters_fig4_349594226

33. Time Series — ARIMA vs. SARIMA vs. LSTM
URL: https://towardsdatascience.com/time-series-arima-vs-sarima-vs-lstm-hands-on-tutorial-bd5630298da3
34. Python
URL: https://www.python.org/
35. Weather Dataset
URL: https://www.kaggle.com/datasets/swatikhedekar/python-project-on-weather-dataset
36. Energy Consumption Time Series Dataset
URL: https://www.kaggle.com/datasets/vitthalmadane/energy-consumption-time-series-dataset
37. 2022 Russia Ukraine War
URL: https://www.kaggle.com/datasets/piterfm/2022-ukraine-russian-war
38. A. Rotshtein, A. Yosef, T. Neskorodeva, D. Katielnikov. Fuzzy Cognitive Map and Mean Square Method in Empirical Modeling : Application in Economics. Expert Systems with Applications, 2024.
39. T. Yu, Q. Gan, G. Feng. Modeling time series. PeerJ Computer Science, 2021.
40. A. Rotshtein. INTELLIGENT EMPIRICAL MODELING: FUZZINESS. CHAOS. CATASTROPHES, 2024.

ДОДАТОК А

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from scipy.special import expit as sigmoid
import cvxpy as cp
from helpers import determine_dimensions, determine_time_delay, fun_trimf, fun_trim_rec, granularity_inclusion, fun_RMSE

# csv = pd.read_csv('russia_losses_equipment.csv')
# cumulative_values = csv['APC'].values.flatten()[::-1]
# time_series = np.diff(cumulative_values)

csv = pd.read_csv('KwhConsumptionBlower78_1.csv')
time_series = csv['Consumption'].values.flatten()

# csv = pd.read_csv('Weather Data.csv')
# time_series = csv['Press_kPa'].values.flatten()[:2000]

determine_time_delay(time_series)
determine_dimensions(time_series)

tau = 7
dimensions = 8

def reconstruct_phase_space(time_series, tau, d):
    n = len(time_series)
    Y = np.array([time_series[i: i + d * tau: tau] for i in range(n - d * tau)])
    return Y
```

```
phase_space = reconstruct_phase_space(time_series, tau, dimensions)

plt.figure(figsize=(8, 6))
ax = plt.axes(projection='3d')
ax.plot3D(phase_space[:, 0], phase_space[:, 1], phase_space[:, 2])
plt.title('Phase Space Reconstruction')
plt.show()

def lyapunov_exponent(time_series, delay, dimension, n_neighbors=10):
    phase_space = reconstruct_phase_space(time_series, delay, dimension)
    N = len(phase_space)

    L_exp = 0
    for i in range(N):
            neighbors = np.argsort(np.sum((phase_space - phase_space[i])**2,
axis=1))[1:n_neighbors+1]
            mean_distance = np.mean(np.sqrt(np.sum((phase_space[neighbors] -
phase_space[i])**2, axis=1)))
        L_exp += np.log(mean_distance)

    L_exp /= (N * delay)
    return L_exp

L_exp = lyapunov_exponent(time_series, tau, dimensions)
print(f"Largest Lyapunov Exponent: {L_exp}")


scaler = MinMaxScaler(feature_range=(0, 1))
data = scaler.fit_transform(time_series.reshape(-1, 1)).flatten()
```

```
train = data[:int(0.8 * len(data))]
test = data[len(train):]

test_row = len(test)
train_row = len(train)
data_row = len(data)
data_rec = np.zeros(data_row)

num_interval = 5
num_c = dimensions
r = 100

defin_center = [0, 0.5, 1]
weight_matrices = np.zeros((num_c, num_c, r))
membership = np.zeros((num_interval, num_c, r))

U_ms, center = fun_trimf(data, num_c, defin_center)
U = U_ms.T

iMax = train_row - num_interval
rand_num = np.random.permutation(iMax)[:r]

for k in range(r):
    membership[:, :, k] = U[rand_num[k]:rand_num[k] + num_interval]

membership[membership <= 0] = 0.0001
membership[membership >= 1] = 0.9999

lambda_ = 5
```

```
b = 0
W_cvx = np.zeros((num_c, num_c))

for k in range(r):
    U1 = membership[:, :, k]
    X = U1[:num_interval-1, :]
    Y = U1[1:num_interval, :]
    D = -(1/lambda_) * np.log((1 - Y) / Y)

    W_cvx = np.zeros((num_c, num_c))

    for i in range(num_c):
        Wx = cp.Variable(num_c)
        objective = cp.Minimize(cp.norm(X @ Wx - D[:, i], 2))
        constraints = [cp.norm(Wx, "inf") <= 1.0]
        prob = cp.Problem(objective, constraints)
        prob.solve()
        W_cvx[:, i] = Wx.value

    weight_matrices[:, :, k] = W_cvx

subFCM_data = np.zeros((data_row, r))
membership_forecast = np.zeros((data_row, num_c, r))

for k in range(r):
    membership_forecast[0, :, k] = U[0]
    for i in range(data_row - 1):
            membership_forecast[i + 1, :, k] = sigmoid(np.dot(U[i - 1, :],
weight_matrices[:, :, k]) * lambda_ + b)
    subFCM_data[:, k] = fun_trim_rec(membership_forecast[:, :, k], center)
```

```
recError = np.zeros(r)
allError = np.zeros((data_row, r))

for j in range(r):
    recError[j] = fun_RMSE(subFCM_data[:, j], data)
    for i in range(data_row):
        allError[i, j] = fun_RMSE(subFCM_data[i, j], data[i])

all_weight = np.zeros((data_row, r))
for i in range(1, data_row):
    all_weight[i, :] = (1. / allError[i, :]) / np.sum(1. / allError[i, :])

data_best = np.zeros(data_row)
data_best[0] = data[0]
data_best[1] = data[1]

for i in range(2, data_row):
        low, up, best = granularity_inclusion(subFCM_data[i], all_weight[i - 1],
allError[i - 1], data[i - 1])
    data_best[i] = best

data_anti = scaler.inverse_transform(data.reshape(-1, 1)).flatten()
data_best = scaler.inverse_transform(data_best.reshape(-1, 1)).flatten()
center1 = scaler.inverse_transform(center.reshape(-1, 1)).flatten()

y = np.array([0, 0.25, 0.5, 1])
y1 = scaler.inverse_transform(y.reshape(-1, 1)).flatten()

plt.figure()
```

```
plt.fill_between(range(train_row + 1, data_row + 1), y1[1], y1[0], color='grey', alpha=0.5)
plt.fill_between(range(train_row + 1, data_row + 1), y1[2], y1[1], color='lightgrey', alpha=0.5)
plt.fill_between(range(train_row + 1, data_row + 1), y1[3], y1[2], color='lightyellow', alpha=0.5)
plt.plot(range(train_row + 1, data_row + 1), data_anti[train_row:], '-k')
plt.plot(range(train_row + 1, data_row + 1), data_best[train_row:], '-r')
plt.legend(['Low', 'Medium', 'High', 'Actual', 'Predicted'])
x_line = range(train_row + 1, data_row + 1)
plt.xlim([train_row + 1, data_row])
plt.show()

plt.figure()
plt.fill_between(range(1, data_row + 1), y1[1], y1[0], color='grey', alpha=0.5)
plt.fill_between(range(1, data_row + 1), y1[2], y1[1], color='lightgrey', alpha=0.5)
plt.fill_between(range(1, data_row + 1), y1[3], y1[2], color='lightyellow', alpha=0.5)
plt.plot(range(1, data_row + 1), data_anti, '-k')
plt.plot(range(train_row + 1, data_row + 1), data_best[train_row:], '-r')
plt.legend(['Low', 'Medium', 'High', 'Actual', 'Predicted'])
x_line = range(train_row + 1, data_row + 1)
plt.xlim([1, data_row])
plt.show()

import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import NearestNeighbors
```

```
# Function to calculate autocorrelation
def autocorrelation(x, lag):
    return np.corrcoef(x[:-lag], x[lag:])[0, 1]

# Determine the time delay (τ) using autocorrelation
def determine_time_delay(time_series):
    lags = range(1, 10)
    autocorrs = [autocorrelation(time_series, lag) for lag in lags]
    tau = lags[np.argmin(autocorrs)]
    print(f'Time delay (τ): {tau}')

    plt.plot(autocorrs)
    plt.title('Autocorrelation function')
    plt.show()

# Function to calculate the proportion of false nearest neighbors
def false_nearest_neighbors(time_series, max_dim, radius=10.0):
    N = len(time_series)
    proportions = []

    for m in range(1, max_dim + 1):
        embedded = np.array([time_series[i:N - m + i] for i in range(m)]).T

        nbrs = NearestNeighbors(n_neighbors=2, algorithm='auto').fit(embedded)
        distances, indices = nbrs.kneighbors(embedded)

        # Compute the proportion of false nearest neighbors
        false_neighbors = 0
        for i in range(N - m):
            distance_ratio = np.abs(
```

```
                time_series[i + m] - time_series[indices[i, 1]]
            ) / distances[i, 1]
            if distance_ratio > radius:
                false_neighbors += 1

        proportion = false_neighbors / (N - m)
        proportions.append(proportion)

    return proportions

def determine_dimensions(time_series):
    # Parameters
    max_dim = 10

    # Calculate the proportion of false nearest neighbors
        proportion_of_false_neighbors = false_nearest_neighbors(time_series,
max_dim)

    # Plotting
    plt.figure(figsize=(8, 5))
     plt.plot(range(1, max_dim + 1), proportion_of_false_neighbors, color='red',
marker='o')
    plt.xlabel(r'$m$')
    plt.ylabel(r'$\delta$')
    plt.title('Proportion of False Neighbors as a Function of Dimension')
    plt.show()


def trimf(x, params):
    a, b, c = params
```

```
    y = np.zeros_like(x, dtype=float)

    # Left side of the triangle
    idx = np.logical_and(a < x, x < b)
    y[idx] = (x[idx] - a) / (b - a)

    # Right side of the triangle
    idx = np.logical_and(b < x, x < c)
    y[idx] = (c - x[idx]) / (c - b)

    # Peak at b
    y[x == b] = 1

    return y

def fun_trimf(data, num, defin=None):
    # Ensure data is a 1D array for consistent handling
    data = np.squeeze(data)  # Remove singleton dimensions if present
    delta = (np.max(data) - np.min(data)) / (num - 1)

    row = len(data)
    U_ms = np.zeros((num, row))  # Correct shape for U_ms
    center = np.zeros(num)

    # Construct first membership function
    U_ms[0, :] = trimf(data, [np.min(data), np.min(data), np.min(data) + delta])
    center[0] = np.min(data)

    # Construct intermediate membership functions
    for i in range(1, num - 1):
```

```
        U_ms[i, :] = trimf(data, [np.min(data) + (i - 1) * delta,
                                  np.min(data) + i * delta,
                                  np.min(data) + (i + 1) * delta])
        center[i] = np.min(data) + i * delta

    # Construct last membership function
    U_ms[num - 1, :] = trimf(data, [np.min(data) + (num - 2) * delta,
                                    np.max(data), np.max(data)])
    center[num - 1] = np.max(data)

    return U_ms, center

def fun_trim_rec(rec_U, rec_center):
    # Get the sizes of the input matrices
    rec_p, U_line = rec_U.shape
    data_rec = np.zeros(rec_p)

    # Loop through each row of rec_U
    for i in range(rec_p):
        sum_up = 0
        sum_down = 0

        # Calculate the sums
        for j in range(U_line):
            sum_up += rec_U[i, j] * rec_center[j].sum()  # Assuming we want the
sum over all elements in rec_center
            sum_down += rec_U[i, j]

        # Avoid division by zero
        if sum_down != 0:
```

```
            data_rec[i] = sum_up / sum_down
        else:
            data_rec[i] = 0  # Handling division by zero case

    return data_rec

def granularity_inclusion(y_data, m_weight, dataError, data_point):
    num = len(y_data)

    # Sort y_data and rearrange weights accordingly
    A = np.sort(y_data)
    A_index = np.argsort(y_data)
    weight = m_weight[A_index]

    # Initialize variables
    obj = np.zeros(int(num * (num - 1) / 2))
    ranges = np.zeros((int(num * (num - 1) / 2), 2))
    alpha = 1
    k = 0

    # Calculate objective values for all intervals
    for i in range(num - 1):
        for ii in range(i + 1, num):
            cov = np.sum(weight[i:ii + 1])
            sp = abs(A[ii] - A[i])
            ranges[k, :] = [A[i], A[ii]]

            # Check if data_point lies within the interval
            if data_point == A[i] or data_point == A[ii] or (A[i] < data_point < A[ii]):
                flag = 1
```

```
        else:
            flag = 0

        # Compute the objective value
        obj[k] = cov * np.exp(-alpha * sp) * flag
        k += 1

# Find the interval with the maximum objective value
position = np.argmax(obj)
low = ranges[position, 0]
up = ranges[position, 1]

# Sort y_data by weights to prepare for weighted average
w_index = np.argsort(m_weight)
B = y_data[w_index]

# Collect data and errors within the selected interval
interval_data = []
interval_error = []
for i in range(num):
    if low <= y_data[i] <= up:
        interval_data.append(y_data[i])
        interval_error.append(dataError[i])

# Convert to numpy arrays
interval_data = np.array(interval_data)
interval_error = np.array(interval_error)

# Compute weights inversely proportional to the errors
interval_weight = (1.0 / interval_error) / np.sum(1.0 / interval_error)
```

```
    # Calculate the best estimate as a weighted average
    best = np.sum(interval_weight * interval_data)

    return low, up, best


def fun_RMSE(X_hat, X_i):
    # Check if the dimensions of X_hat and X_i are the same
    if X_hat.shape != X_i.shape:
        print("error")
        return None

    if not isinstance(X_hat, np.ndarray):
        X_hat = np.array([X_hat])
    if not isinstance(X_i, np.ndarray):
        X_i = np.array([X_i])

    # Calculate the sum of squared differences
    sum_squared_diff = 0
    for i in range(X_hat.shape[0]):  # Iterate over rows
        temp = np.linalg.norm(X_hat[i] - X_i[i]) ** 2
        sum_squared_diff += temp

    # Compute the RMSE
    rmse = np.sqrt(sum_squared_diff / X_hat.shape[0])
    return rmse
```