Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Системотехніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

_____Дослідження та застосування генетичних алгоритмів для прогнозування температури земної поверхні_____
_____
(тема)

Виконав:
студент _2__ курсу, групи _____СПРм-20-1____
_____Василенко А.К._____
(прізвище, ініціали)

Спеціальність ____122 Комп'ютерні науки____
_____
(код і повна назва спеціальності)

Тип програми _____Освітньо-професійна__
(освітньо-професійна або освітньо-наукова)

Освітня програма _____ОПП_Системне____проектування_____
(повна назва освітньої програми)

Керівник _____доц._Панкратов О.В._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ _____Гребеннік І.В._____
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Системотехніки

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма ОПП Системне проектування
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Василенко Альоні Костянтинівні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та застосування генетичних алгоритмів для прогнозування температури земної поверхні

затверджена наказом університету від 8 листопада 2021 р. № 1663 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 грудня 2021 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані відомих наукових проектів щодо розробки систем спостереження погоди, дані статей, результати експериментальних досліджень, вибірка даних про температуру поверхні

4. Перелік питань, що потрібно опрацювати в роботі

1. Аналіз предметної галузі та постановка задачі,
2. Еволюційне набуття нейронних топологій,
3. Аналіз інструментальних засобів,
4. Імітаційне моделювання

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
__Слайди_презентації:_титул,_вступ,_актуальність,_мета,_постановка_задачі,_____
алгоритм_EANT,_RNN,_інструментальні_засоби,_експериментальні дослідження,_____
представлення_результатів,порівняння_результатів,_висновки,_перспективи_розвитку__
_____
_____
_____
_____
_____
_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	доц. Панкратов О.В.		

1. КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Отримання завдання кваліфікаційної роботи	8.11.2021	виконано
2.	Аналіз предметної галузі і постановка завдання	9.11.2021-12.11.2021	виконано
3.	Дослідження методів та технологій	13.11.2021-14.11.2021	виконано
4.	Аналіз існуючих реалізацій	15.11.2021-17.11.2021	виконано
5.	Розробка методу рішення поставленої проблеми	18.11.2021-21.11.2021	виконано
6.	Створення імітаційної моделі	22.11.2021-27.11.2021	виконано
7.	Тестування і опрацювання імітаційної моделі	28.11.2021-30.11.2021	виконано
8.	Оформлення пояснювальної записки	1.12.2021-11.12.2021	виконано
9.	Оформлення графічних матеріалів	12.12.2021-14.12.2021	виконано
10.	Попередній захист	16.12.2021	виконано
11.	Захист перед ЕК	18.12.2021	

Дата видачі завдання __8___ ___листопада__2021 р.

Студент _____
(підпис)

Керівник роботи _____ _____доц. Панкратов О.В.___
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Записка пояснювальна: 59 с., 13 рис., 1 табл., 2 дод., 19 джерел.

ГЛУБОКЕ НАВЧАННЯ, ЕВОЛЮЦІЙНІ АЛГОРИТМИ, НЕЙРОЕВОЛЮЦІЯ, ЧАСОВІ РЯДИ, EANT, INDIRECT ENCODING, KERAS, PYTHON, REINFORCMENT LEARNING, RNN

У кваліфікаційній роботі розглядається вирішення проблеми прогнозування температури земної поверхні за допомогою технологій штучного інтелекту, а саме нейронних мереж і еволюційних алгоритмів. Поставлена задача полягає в тому, щоб розробити програмну систему, яка буде з використанням архіва метеоролгічних даних робити передбачення щодо параметрів погоди у заданій точці земної поверхні в обраний проміжок часу.

Для вирішення поставленої задачі було проаналізовано математичну модель задачі, існуючі методи та алгоритми для вирішення подібних задач, переваги і недоліки генетичних алгоритмів і використання штучних нейроних мереж для знаходження оптимального розв’язку цієї задачі. В якості інструменту використовуються рекурентні нейронні мережі, що показують гарні результати у задачах з часовими рядами, і еволюційний алгоритм EANT для навчання мережі і створення її ефективної структури.

Також в цій роботі описується процес імітаційного моделювання, який включає в себе створення, навчання і використання таких нейроних мереж. Проводиться порівняльний аналіз точності моделей нейроних мереж для поставленої задачі серед обраних алгоритмів.

У результаті було показано, що генетичні алгоритми, як метаевристичні алгоритми, дають доволі швидкий і якісний розв’язок задачі, досягаючи у порівнянні з іншими алгоритмами переваги на 7% у точності передбачення.

РЕФЕРАТ

Пояснительная записка: 59 с., 13 рис., 1 табл., 2 прил., 19 источников.

ВРЕМЕННЫЕ РЯДЫ, ГЛУБОКОЕ ОБУЧЕНИЕ, НЕЙРОЭВОЛЮЦИЯ, ЭВОЛЮЦИОННЫЕ АЛГОРИТМЫ, EANT, INDIRECT ENCODING, PYTHON, REINFORCMENT LEARNING, RNN

В квалификационной работе рассматривается решение проблемы прогнозирования температуры земной поверхности с помощью технологий искусственного интеллекта, а именно нейронных сетей и эволюционных алгоритмов. Поставленная задача состоит в том, чтобы разработать программную систему, которая будет с использованием архива метеорологических данных делать предсказание параметров погоды в заданной точке земной поверхности в выбранный промежуток времени.

Для решения поставленной задачи была проанализирована математическая модель задачи, алгоритмы решения подобных задач, преимущества и недостатки генетических алгоритмов и нейронных сетей для нахождения оптимального решения этой задачи. В качестве инструмента используются рекуррентные нейронные сети, показывающие хорошие результаты в задачах с временными рядами, и эволюционный алгоритм EANT для обучения сети и создания ее эффективной структуры.

Также в этой работе описывается процесс имитационного моделирования, включающий в себя создание, обучение и использование таких нейронных сетей. Проводится сравнительный анализ точности моделей для поставленной задачи среди выбранных алгоритмов.

В результате было показано, что генетические алгоритмы, как метаэвристические алгоритмы, дают достаточно быстрое и качественное решение задачи, достигая по сравнению с другими алгоритмами преимущества на 7% в точности предсказания.

ABSTRACT

Explanatory note: 59 pages, 13 figures, 1 table, 2 appendixes, 19 sources.

DEEP LEARNING, EVOLUTIONARY ALGORITHMS, INDIRECT ENCODING, KERAS, EANT, NEUROEVOLUTION, PYTHON, REINFORCMENT LEARNING, RNN, TIME SERIES

This work describes the solution of the problem of predicting the temperature of the earth's surface using artificial intelligence technologies, namely neural networks and evolutionary algorithms. The problem is formulated as follows: develop a software system that will use the meteorological data archive to predict weather parameters at a given point on the earth's surface at a selected time interval.

To solve the problem, its mathematical model, existing algorithms for solving such problems, the advantages and disadvantages of genetic algorithms and the adaptation of artificial neural networks to find the optimal solution to this problem were analyzed. Recurrent neural networks, which show good results in time series problems, and the evolutionary algorithm EANT for learning the network and creating its efficient structure are used as tools.

This paper also describes the process of simulation modeling, which includes the creation, training and use of such neural networks. A comparative analysis of the accuracy of neural network models for the problem among the selected algorithms is presented.

As a result, it was shown that genetic algorithms, as metaheuristic algorithms, provide a fairly fast and high-quality solution to the problem, reaching an advantage of 7% in prediction accuracy compared to other algorithms.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ГА – генетичний алгоритм;

ЗНМ – згорткова нейронна мережа;

МПВ – марківський процес вирішування;

НЕ – нейроеволюція;

ШН – штучний нейрон;

ШНМ – штучна нейронна мережа;

ANN – artificial neural network;

EANT - evolutionary acquisition of neural topologies;

GDS – generative and developmental systems;

RNN – recurent neural network;

ReLU – rectified linear unit;

TWEANNs – topology and weight evolving artificial neural networks.

ВСТУП

Задача прогнозування погоди була актуальною на протязі усієї історії людства. Від поточних умов навколишнього середовища залежало і ведення господарства, і початок будівництва, і організація заходів. Однак, погода є доволі складним фактором для прогнозування через велику кількість різноманітних чинників, у тому числі стохастичних. Перші метеорологічні прогнозування були засновані на простому аналізу історії показників за попередні роки, однак сучасне життя потребує більш досконалих методів обробки і аналізу даних. Одним з провідних сучасних підходів до проблеми прогнозування погоди є використання штучних нейронних мереж. Ця технологія вже показала результати з прогнозування у різних галузях діяльності, що перевершують по швидкості і точності прогнози, зроблені людиною. При цьому, як чисто математичний метод, штучні нейронні мережі піддаються досить легкій оптимізації і дозволяють отримати велику кількість необхідних даних у реальному часі або на декілька років вперед. Однак, при використанні штучних нейронних мереж завжди існувала головна проблема: обрання необхідної архітектури. Якщо проблему з вибором типу штучних нейронних мереж людина вже давно вирішала на основі досвіду використання, то питання про оптимальну кількість шарів, нейронів і їхніх ваг досить не має відповіді. Одним з дуже цікавих і прогресивних методів є використання генетичних алгоритмів, що натхнені самою природою. Вони дозволяють знайти ефективну структуру для конкретної задачі за достатню кількість виділених ресурсів для обчислення. Це призводить до того, що людина хоч і залишається оператором для налаштування таких мереж, однак вона вирішує питання стосовно автоматичної побудови архітектури, залишаючи усе інше для обчислювальної машини. Такий підхід дозволяє ітеративно підбирати для кожного класу проблем найбільш придатні моделі, що показують дуже гарні результати у порівнянні з іншими підходами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Методи прогнозування погоди

Існують три типи методів для вирішення задачі прогнозування температури і погоди, які представлені далі.

Синоптичний метод складання прогнозів погоди ґрунтується на аналізі карт погоди. Сутність цього методу полягає в одночасному огляді стану атмосфери на великій території, що дозволяє визначити характер розвитку атмосферних процесів і подальшу найбільш ймовірну зміну погодних умов в районі. Здійснюється такий огляд за допомогою карт погоди, на які наносяться дані метеорологічних спостережень на різних висотах, а також на поверхні землі, що роблять одночасно за однією програмою в різних точках земної кулі. На основі докладного аналізу цих карт синоптик визначає подальші умови розвитку атмосферних процесів у певний період часу та розраховує характеристики метеоелементів – температуру, вітер, хмарність, опади тощо.

Чисельні (гідродинамічні) методи прогнозу погоди ґрунтуються на математичному вирішенні системи повних рівнянь гідродинаміки та одержання прогностичних полів тиску, температури на певні проміжки часу. Обчислювальні центри Москви, Вашингтону, Токіо і Рейдингу (Європейський прогностичний центр) використовують різноманітні схеми розвитку великомасштабних атмосферних процесів. Точність чисельних прогнозів залежить від швидкості розрахунку обчислювальних систем, кількості та якості інформації, що надходить з метеостанцій. Що більше даних, то точніше розрахунок.

Статистичні методи прогнозу дозволяють за минулим і сьогоденням атмосфери спрогнозувати на певний період часу стан погоди, тобто, передбачити зміни різних метеоелементів у майбутньому. До цих методів відносяться штучні нейронні мережі (ШНМ). Основа цього методу полягає

в узагальненні усіх отриманих даних з вибірки і навчанні ШНМ таким чином, що при отриманні у вихідному шарі початкових метеоданих (координати, температура, швидкість вітру, кількість опадів тощо), ШНМ віддавала би результат щодо подальшого значення температури на обраній поверхні, заснований на математичній обробці даних. Цей метод прогнозування засновується лише на використанні минулих даних і доволі чутливий до можливих «викидів», тож якнайкраще показує тенденцію зміни температури за деякий проміжок у майбутньому.

1.2 Штучна нейронна мережа і її складові

ШНМ є обчислювальними системами, натхненними біологічними нейронними мережами, які утворюють мозок тварин.

ШНМ заснована на сукупності пов’язаних одиниць або вузлів, які називаються штучними нейронами, які вільно моделюють нейрони в біологічному мозку. Кожне з’єднання, як і синапси в біологічному мозку, може передавати сигнал іншим нейронам. Штучний нейрон отримує сигнал, потім обробляє його і може сигналізувати підключеним до нього нейронам. «Сигнал» при з’єднанні є дійсним числом, а вихід кожного нейрона обчислюється за допомогою деякої нелінійної функції суми його вхідних даних [4]. З’єднання називаються ребрами. Нейрони та ребра зазвичай мають вагу, яка коригується у процесі навчання. Вага збільшує або зменшує силу сигналу при з’єднанні. Нейрони можуть мати такий поріг, що сигнал надсилається лише в тому випадку, якщо сукупний сигнал перевищує цей поріг. Як правило, нейрони об’єднуються в шари. Різні шари можуть виконувати різні перетворення на своїх входах. Сигнали поширюються від першого шару (вхідного шару) до останнього шару (вихідного шару), навіть, можливо, після багаторазового проходження шарів. Загальний приклад штучного нейрона представлено на рисунку 1.1.

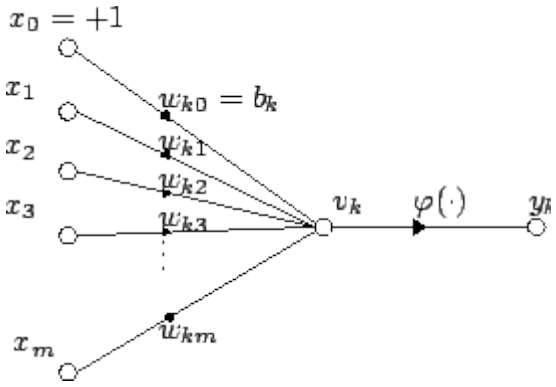


Рисунок 1.1 – Схема штучного нейрону

ШНМ перетворилися на широке сімейство методів, які просунули сучасний рівень техніки в багатьох областях. Найпростіші типи мають один або кілька статичних компонентів, включаючи кількість одиниць ШН, кількість шарів, вагу одиниць ШН і топологію. Динамічні типи ШНМ дозволяють одному або більше з ШН розвиватися шляхом навчання. Вони набагато складніші, але можуть скоротити періоди навчання та дати кращі результати. Деякі типи ШНМ дозволяють або вимагають, щоб навчання проходило під наглядом «вчителя», тоді як інші працюють незалежно [6]. Деякі типи ШНМ працюють виключно в апаратному забезпеченні, а інші є суто програмними та працюють на комп'ютерах загального призначення.

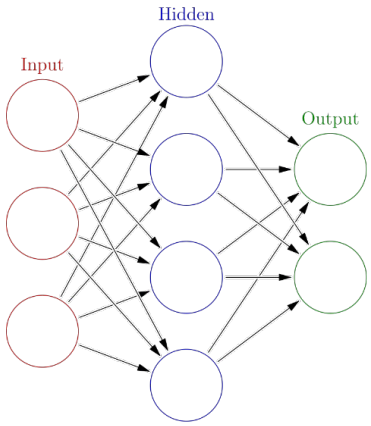


Рисунок 1.2 – Найпростіша штучна мережа

Нейронні мережі навчаються шляхом обробки прикладів, кожен з яких містить відомі «вхідні дані» та «результат», утворюючи зважені за ймовірністю асоціації між ними, які зберігаються в структурі даних самої мережі. Навчання нейронної мережі на прикладах зазвичай проводиться шляхом визначення різниці між обробленим виходом мережі (часто передбаченням) і цільовим виходом. Ця різниця і є помилкою. Потім мережа коригує свої зважені асоціації відповідно до правила навчання та з використанням цього значення помилки. Послідовні коригування змусять нейронну мережу змінювати вихід, який стане все більше подібний до цільового. Після достатньої кількості цих налаштувань навчання може бути припинено за певними критеріями. Це відомо як навчання з вчителем.

Такі системи «вчаться» виконувати завдання, розглядаючи приклади, як правило, без програмування специфічними для завдання правилами. Наприклад, під час розпізнавання зображень вони можуть навчитися ідентифікувати зображення, які містять котів, аналізуючи приклади зображень, які вручну позначено як «кіт» або «без кота», і використовуючи результати, щоб ідентифікувати котів на інших зображеннях. Вони роблять це без будь-яких попередніх знань про кішок, наприклад, що у них є шерсть, хвости, вуса та котячі мордочки. Натомість вони автоматично генерують ідентифікаційні характеристики на основі прикладів, які вони обробляють.

В стандартній конфігурації ШНМ вхідні сигнали множаться на певні ваги і обробляються адаптивним суматором, після чого цими даними займається активаційна функція, результат якої передається на вхід наступному шару (ibid.). Як було сказано, це нелінійна функція, що обчислює вихідний сигнал і вибір якої дуже впливає на налаштування ШНМ для певних задач. Найбільш поширеніші активаційні функції представлено на рисунку 1.3.

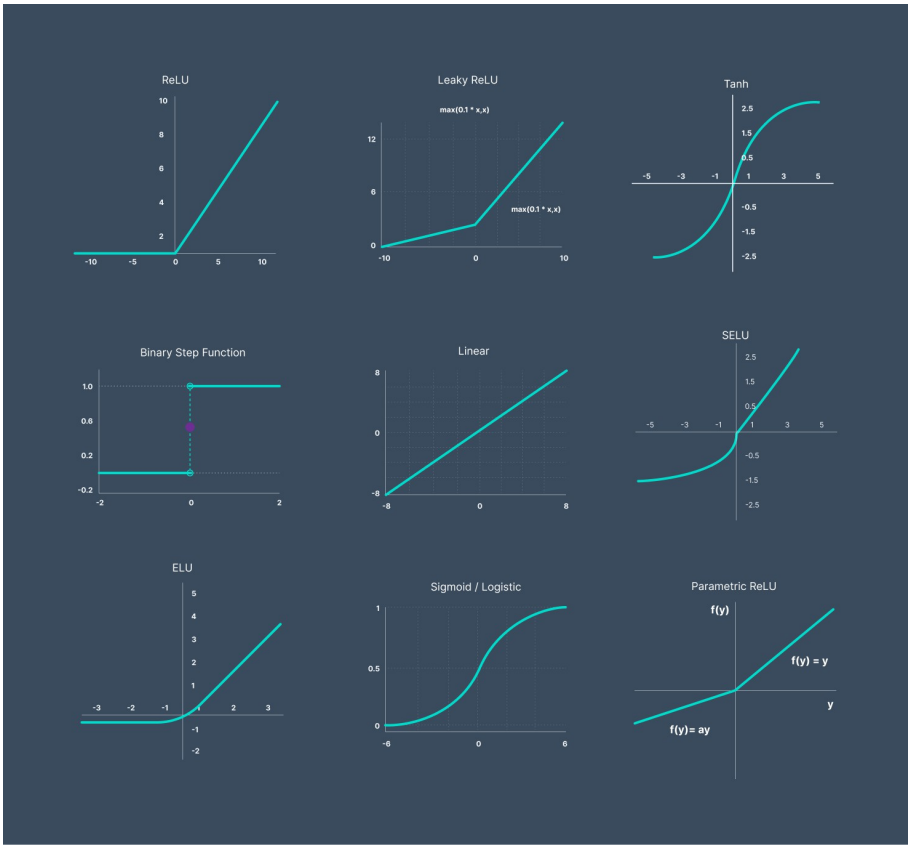


Рисунок 1.3 – Активаційні функції

1.3 Навчання з підкріпленням

Навчання з підкріпленням це галузь машинного навчання, натхнена біхевіористською психологією, що вивчає питання про те, які дії повинні виконувати програмні агенти в певному середовищі задля максимізації деякого уявлення про сукупну винагороду. Навчання з підкріпленням відрізняється від стандартного навчання з учителем тим, що пари правильних входів/виходів ніколи не представляються, а недостатньо оптимальні дії явно не виправляються. Крім того, є акцент на інтерактивній продуктивності, який включає знаходження балансу між дослідженням та використанням поточного знання [9].

Середовище зазвичай виражається у формі процесу прийняття рішень Маркова, оскільки багато алгоритмів навчання з підкріпленням для цього контексту використовують методи динамічного програмування.

Основна відмінність між класичними методами динамічного програмування та алгоритмами навчання з підкріпленням полягає в тому, що останні не передбачають знання точної математичної моделі процесу Маркова і націлені на великі процеси прийняття рішень, де точні методи стають нездійсненними.

Потужним навчання з підкріпленням роблять дві складові: використання зразків для оптимізації продуктивності, та застосування наближень функцій, що дозволяє мати справу з великими середовищами. Завдяки цим двом складовим навчання з підкріпленням можливо застосовувати у великих середовищах в будь-яких із наступних ситуацій:

- модель середовища є відомою, але аналітичний розв’язок відсутній;
- задано лише імітаційну модель середовища (предмет оптимізації на основі імітації);
- єдиним способом збирання інформації про середовище є взаємодія з ним.

Метою даної роботи є використання еволюційних стратегій для розвитку тополгії і ваг ШНМ. Це вимагає використовування даного підходу для навчання, оскільки еволюційний процес завжди спирається на функцію пристововуваності. У процесі розвитку немає поняття «вхідні-вихидні дані». Залежно від результатів перевірки на практиці оцінюється деякий показник у числовому вигляді, який має екстремум, до якого повинна прагнути модель ШНМ. У наступних розділах це буде описано більш детально.

1.4 Еволюційні алгоритми

Еволюційні обчислення — це сімейство алгоритмів для глобальної оптимізації, натхненних біологічною еволюцією, а також підобласть штучного інтелекту та м’яких обчислень, що вивчає ці алгоритми. У технічному плані вони являють собою сімейство популяційних

розв’язувачів проблем методом проб і помилок з метаевристичним або стохастичним характером оптимізації.

При еволюційних обчисленнях генерується і ітеративно оновлюється початковий набір рішень-кандидатів. Кожне нове покоління створюється шляхом стохастичного видалення менш бажаних рішень і внесення невеликих випадкових змін. У біологічній термінології популяція результатів піддається природному відбору (або штучному добору) і мутації. В результаті популяція буде поступово розвиватися до збільшення пристосованості, у цьому випадку обраної функції пристосованості алгоритму [12].

Методи еволюційного обчислення можуть створювати високооптимізовані рішення для широкого діапазону проблем, що робить їх популярними в інформатиці. Існує багато варіантів і розширень, які підходять для більш конкретних сімейств проблем і структур даних. Еволюційні обчислення також іноді використовуються в еволюційній біології як експериментальна процедура «in silico» для вивчення загальних аспектів еволюційних процесів.

У еволюційному обчисленні використовують генетичні алгоритми, еволюційне програмування, еволюційні стратегії, системи класифікаторів, генетичне програмування тощо. Усе це використовується для моделювання базових положень теорії біологічної еволюції — процесів відбору, мутації і відтворення.

Популяція складається з множини агентів (індивідів). Поведінка агентів визначається довкіллям. Така популяція еволюціонує відповідно до правил відбору і цільової функції, що задається довкіллям. Таким чином, кожному агентові популяції призначається значення його пристосовуваності в довкіллі (ibid.). Розмножуються лише найпридатніші види.

Слід приділити увагу генетичним алгоритмам (ГА). Вони зазвичай використовуються для створення високоякісних рішень проблем

оптимізації та пошуку, покладаючись на біологічно натхненні оператори, такі як мутація, кросовер і відбір. У ГА популяція варіантів рішень (так званих індивідів, істот або фенотипів) оптимізаційної проблеми розвивається до кращих рішень [13]. Кожне рішення-кандидат має набір властивостей (його хромосоми або генотип), які можуть бути мутовані та змінені; традиційно рішення представлені в двійковому вигляді (рядки з 0 і 1), але можливі й інші кодування.

Еволюція зазвичай починається з популяції випадково згенерованих індивідів і є ітераційним процесом, при цьому популяція на кожній ітерації називається поколінням. У кожному поколінні оцінюється придатність кожної особини в популяції; придатність - це зазвичай значення цільової функції в оптимізаційній задачі, що вирішується. Більш придатні особини стохастично відбираються з поточної популяції, і геном кожної особи модифікується (рекомбінується і, можливо, випадково мутується) для формування нового покоління. Нове покоління варіантів рішень використовується на наступній ітерації алгоритму. Зазвичай алгоритм припиняє роботу, коли або створено максимальну кількість поколінь, або досягнуто задовільного рівня придатності для популяції.

Типовий ГА вимагає генетичного представлення області рішення і функції придатності для оцінки області рішення. Стандартне представлення кожного рішення-кандидата є масивом бітів (також званим набором бітів або бітовим рядком). Масиви інших типів і структур можна використовувати, по суті, таким же чином. Основна властивість, яка робить ці генетичні представлення зручними, полягає в тому, що їх частини легко вирівнюються завдяки фіксованому розміру, що полегшує прості операції кросинговеру. Також можуть використовуватися представлення змінної довжини, але в цьому випадку реалізація кросовера є більш складною [3]. Деревоподібні представлення досліджуються в генетичному програмуванні, а представлення у формі графів досліджуються в еволюційному програмуванні; поєднання лінійних

хромосом і дерев досліджується в програмуванні експресії генів.

Після визначення генетичного представлення та функції придатності ГА переходить до ініціалізації популяції рішень, а потім до її покращення шляхом повторюваного застосування операторів мутації, кросинговеру, інверсії та відбору.

Розмір популяції залежить від природи проблеми, але зазвичай містить кілька сотень або тисяч можливих рішень. Часто початкова сукупність генерується випадковим чином, що дозволяє отримати весь спектр можливих рішень (простір пошуку). Іноді рішення можуть бути «засіяні» в областях, де, ймовірно, будуть знайдені оптимальні рішення. Протягом кожного наступного покоління відбирається частина існуючої популяції для виведення нового покоління. Індивідуальні рішення вибираються за допомогою процесу, що ґрунтується на придатності, де, як правило, з більшою ймовірністю будуть обрані кращі рішення (виміряні функцією придатності). Певні методи відбору оцінюють придатність кожного рішення і переважно вибирають найкращі рішення. Інші методи оцінюють лише випадкову вибірку сукупності, оскільки перший процес може зайняти дуже багато часу [8].

Функція придатності визначається за генетичним представленням і вимірює якість представленого рішення. Функція придатності завжди залежить від проблеми. Наприклад, у задачі про ранець потрібно максимізувати загальну вартість предметів, які можна помістити в ранець певної фіксованої місткості. Подання рішення може бути масивом бітів, де кожен біт представляє інший об’єкт, а значення біта (0 або 1) показує, чи знаходиться об’єкт у ранці. Не кожне таке представлення справедливе, оскільки розміри предметів можуть перевищувати місткість ранця. Придатність рішення — це сума значень усіх об’єктів у рюкзаку, якщо представлення вірна, або 0 — в іншому випадку.

У деяких задачах важко або навіть неможливо визначити функцію придатності; у цих випадках можна використовувати моделювання для

визначення значення функції пристосованості фенотипу (наприклад, обчислювальна гідродинаміка використовується для визначення опору повітря транспортного засобу, форма якого закодована як фенотип), або навіть використовувати інтерактивні генетичні алгоритми.

Наступним кроком є створення популяції другого покоління рішень із тих, які вибрано за допомогою комбінації генетичних операторів: кросовер (також званий рекомбінацією) та мутація.

Для кожного нового рішення, яке буде створено, обирається пара «батьківських» рішень для розведення з попередньо вибраного набору. З використанням вищевказаних методів кросинговеру та мутації створюється «дочірнє» рішення, яке зазвичай має багато характеристик своїх «батьків». Для кожної нової «дитини» обираються нові «батьки», і цей процес триває доти, поки не буде створено нову сукупність рішень відповідного розміру. Хоча методи розмноження, засновані на використанні двох «батьків», більше «надихаються біологією», деякі дослідження показують, що більше двох «батьків» генерують хромосоми вищої якості.

Ці процеси в кінцевому підсумку призводять до популяції хромосом наступного покоління, яка відрізняється від початкового покоління. Як правило, середня пристосованість підвищується завдяки цій процедурі для популяції, оскільки для розведення відбираються лише найкращі організми з першого покоління, а також невелика частка менш придатних розчинів. Ці менш відповідні рішення забезпечують генетичне різноманіття в генетичному фонді батьків і, отже, забезпечують генетичну різноманітність наступного покоління дітей.

Хоча кросовер і мутація відомі як основні генетичні оператори, у генетичних алгоритмах можна використовувати інші оператори, такі як перегрупування, «колонізація-вимирання» або міграція [14].

Варто налаштувати такі параметри, як ймовірність мутації, ймовірність кросинговеру та розмір популяції, щоб знайти розумні

налаштування для класу проблеми, над яким проводиться дослідження. Дуже малий рівень мутації може призвести до генетичного дрейфу (який за своєю природою не є ергодічним). Занадто висока швидкість рекомбінації може призвести до передчасної конвергенції генетичного алгоритму. Занадто високий рівень мутації може призвести до втрати хороших рішень, якщо не використовується елітарний відбір. Адекватний розмір популяції забезпечує достатнє генетичне різноманіття для розглянутої проблеми, але може призвести до втрати обчислювальних ресурсів, якщо встановлено значення більше, ніж потрібно.

На додаток до основних операторів, наведених вище, можна використовувати інші евристики, щоб зробити обчислення швидшим або надійнішим. Евристика видів штрафує перехід між занадто схожими рішеннями-кандидатами; це сприяє різноманітності популяції та допомагає запобігти передчасному зближенню до менш оптимального рішення.

Процес генерації повторюється до тих пір, поки не буде досягнута умова припинення. Загальні умови завершення: знайдено рішення, яке задовольняє мінімальним критеріям; фіксована кількість досягнутих поколінь; використано виділений бюджет (час обчислень/гроші); придатність рішення з найвищим рейтингом досягла піку, так що послідовні ітерації більше не дають кращого результату; ручна перевірка; комбінації перерахованого вище [12].

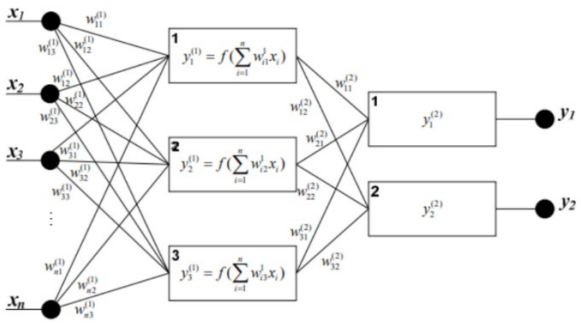


Рисунок 1.4 – Двошарова нейронна мережа з ваговими коефіцієнтами – генами хромосом

1.5 Постановка завдання дослідження

Формально завдання дослідження зводиться до аналізу існуючих видів і архітектур мереж та методів їх навчання, створенню штучної нейронної мережі для метеорологічного прогнозування, проведенню її навчання та оптимізації з використанням генетичного алгоритму (EANT). Ціллю завдання є рішення проблеми прогнозування температури поверхні в обраний проміжок часу, ґрунтуючись на минулих показниках. Потрібно дослідити обрану штучну нейрону мережу для знаходження кількості вхідних нейронів, створити систему для імітації задачі прогнозування температури поверхні, реалізувати генетичний алгоритм у парі зі штучною нейронною мережею, навчити цю мережу на основі навчання з підкріпленням та проходу багатьох поколінь, протестувати отриману модель в різних умовах, проаналізувати отримані результати та порівняти з існуючими реалізаціями.

Для досягнення поставленої мети необхідно розглянути наступні питання:

- провести аналіз існуючих методів еволюційних алгоритмів для вирішення задачі навчання та побудови архітектури штучної нейронної мережі;

- розробити метод і нейронну мережу для створення системи прогнозування температури на поверхні;

- провести імітаційне моделювання та порівняльний аналіз розробленої штучної нейронної системи з існуючими на даний момент іншими системами для описаної задачі.

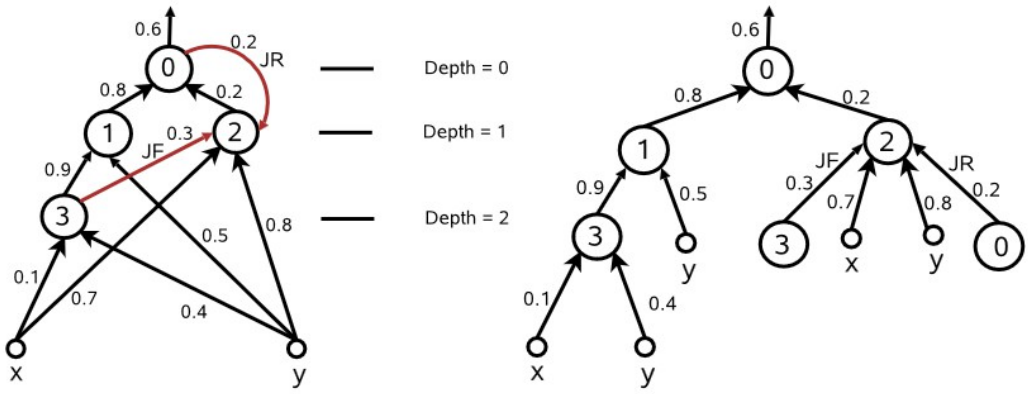
Наприкінці ми маємо впевнитися, що даний підхід дозволяє отримати кращі результати у порівнянні з існуючими методами і реалізаціями, і може бути придатний до подальшого розвитку і дослідження для вирішення різних областей задач с подібними мережі.

2 НЕЙРОЗВОЛЮЦІЯ ТОПОЛОГІЙ ШТУЧНИХ НЕЙРОННІХ МЕРЕЖ

2.1 Кодування структури

Гнучке кодування дозволяє розробити ефективний еволюційний метод, який може розвивати як структуру, так і ваги нейронних мереж. Геном в EANT створюється з урахуванням цього факту. Геном в EANT – це лінійний геном генів (вузлів), які можуть приймати різні форми (алелі). Форми, які може приймати ген, можуть бути або нейроном, або входом до нейронної мережі, або геном перемички, що з'єднує два нейрони. Гени перемички вводяться шляхом структурної мутації на шляху еволюції. Вони кодують прямі або зворотні з'єднання. Ген перемички, що кодує прямий зв'язок, представляє з'єднання, що починається від нейрона на більшій глибині і закінчується нейроном на меншій глибині. З іншого боку, ген перемички, що кодує зворотній зв'язок, являє собою з'єднання між нейронами однакової глибини або зв'язок, що починається від нейрона на меншій глибині і закінчується нейроном на більшій глибині. Кожен вузол у лінійному геномі має пов'язану з ним вагу. Вага кодує синаптичну силу зв'язку між вузлом, кодованим геном і нейроном, з яким він з'єднаний. Більше того, кожен вузол може зберегти результати поточного обчислення. Це корисно, оскільки результати сигналів у зворотніх зв'язках доступні на наступному кроці обчислення. На додаток до синаптичної ваги вузол нейрона має унікальний глобальний ідентифікаційний номер і кількість вхідних з'єднань з ним. Вузол перемички також додатково має глобальний ідентифікаційний номер, який показує нейрон, до якого він підключений. Приклад лінійного геному, що кодує нейронну мережу, показаний на рисунку 2.1. У лінійному геномі N означає нейрон, I — вхід у нейронну мережу, JF — пряме з'єднання, а JR — зворотнє з'єднання. Цифри біля N представляють глобальні ідентифікаційні номери нейронів, а x або y

представляють вхідні дані, закодовані вхідним геном (вузлом).



N 0 W=0.6	N 1 W=0.8	N 3 W=0.9	I x W=0.1	I y W=0.4	I y W=0.5	N 2 W=0.2	JF 3 W=0.3	I x W=0.7	I y W=0.8	JR 0 W=0.2

(c) Corresponding Linear Genome

Рисунок 2.1 – Приклад кодування нейронної мережі за допомогою лінійного геному

Лінійний геном EANT можна інтерпретувати як лінійну програму, яка кодує програму на основі дерева, якщо припустити, що всі входи нейронної мережі та всі з’єднання є параметрами, а нейрони – функціями. Програму на основі дерева (нейронну мережу) можна зберігати в масиві (лінійний геном), де структура дерева (топологія мережі) неявно кодується в порядку елементів масиву [14]. Це призводить до компактного кодування нейронних структур за допомогою лінійних геномів.

Лінійний геном має деякі цікаві властивості, що робить його корисним для еволюції структури нейронних мереж. Якщо призначити цілі значення кожному з вузлів лінійного геному так, що ціле значення представляє різницю між кількістю виходів вузла, яка завжди дорівнює одиниці, і кількістю входів у вузол, можна легко побачити, що сума цілих

значень дорівнює кількості виходів нейронної мережі, закодованих лінійним геномом. Ціле значення одиниці призначається всім нейронам і прямим/зворотимх вузлам, оскільки нейрони та вузли розглядаються як джерело сигналів. Вузлу нейрона призначається ціле значення , де - кількість входів у нейрон. Підмережу, яка починається з будь-якого вузла нейрона в лінійному геномі, можна легко ідентифікувати за допомогою правила, яке стверджує, що сума цілих значень, призначених вузлам між початковим вузлом нейрона та кінцевим вузлом підмережі включно, дорівнює одиниці [11].

На додаток до компактного представлення нейронних структур, лінійний геном пропонує можливість оцінки нейронної мережі, закодованої лінійним геномом, без її декодування. Процес оцінки лінійного геному без декодування закодованої ним нейронної мережі виконується наступним чином. Ми починаємо з самого правого елементу лінійного геному, а потім рухаємося ліворуч, обчислюючи вихідні дані елементів. Якщо поточний елемент є вхідним, ми вставляємо його поточне значення і пов’язану з ним вагу в стек. Якщо поточний елемент є нейроном, ми витягуємо n значень з пов’язаними з ними вагами зі стека і переносимо результат обчислення з вагою, пов’язаною із вузлом нейрона, до стека. Якщо поточний елемент є зворотнім вузлом, ми отримуємо останнє значення вузла нейрона, глобальний ідентифікаційний номер якого такий самий, як і зворотнього вузла. Потім ми поміщаємо отримане значення з вагою, пов’язаною з вузлом, в стек. Якщо поточний елемент є прямим вузлом, спочатку ми копіюємо сублінійний геном (підмережу), починаючи з нейрона, глобальний ідентифікаційний номер якого такий самий, як і у прямого вузла. Ми обчислюємо вихід сублінійного геному так само, як і лінійного геному. Нарешті, ми переносимо результат обчислення з вагою, пов’язаним із прямим вузлом, до стека. Після повного проходження генома справа наліво ми витягуємо отримані значення зі стеку. Кількість отриманих значень така ж, як і кількість виходів

нейронної мережі, закодованих лінійним геномом. Приклад оцінки лінійного геному показаний на рисунку 2.2. Для цього прикладу поточні значення вхідних даних нейронної мережі, x і y, встановлені в 1. У прикладі всі нейрони мають лінійну функцію активації у вигляді , де — зважена лінійна комбінація входів до нейрона. Числа над лінійним геномом показують стан стека після обчислення вихідних даних вузла. Числа в дужках – це ваги, пов’язані з вузлами. Числа в квадратних дужках під лінійним геномом показують цілі значення, призначені вузлам лінійного геному. Зауважимо, що сума цілих значень дорівнює одному, що показує, що нейронна мережа, закодована лінійним геномом, має лише один вихід. Затінені вузли утворюють підмережу. Зауважимо також, що сума цілих значень, призначених підмережі, завжди дорівнює одиниці.

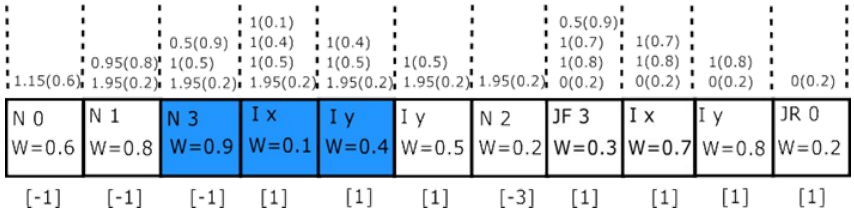


Рисунок 2.2 – Приклад оцінки лінійного геному без декодування нейронної мережі

Оцінка лінійного геному, розглянута вище, еквівалентна оцінці декодованої нейронної мережі, представленої геномом, де активація нейрона мережі задається як

. (1)

У рівнянні — функція активації нейрона, а — кількість вхідних з’єднань із нейроном. Кількість прямих і зворотніх з'єднань з нейроном дорівнює і відповідно [12].

Оцінка лінійного геному тісно пов’язана з виконанням лінійної програми за допомогою постфіксної нотації. У генетичному кодуванні

операнди (входи) стоять перед оператором (нейронна мережа).

2.2. Алгоритм EANT

EANT зазвичай починається з мінімальної початкової структури. Мінімальна мережа не має прихованих шарів або зворотні з'єднання, лише один нейрон на кожен вихід, підключений до деяких або всіх нейронів. EANT поступово розвиває ці прості початкові структури, надалі з використанням структурних і параметричних еволюційних алгоритмів, розглянутих нижче. У більшому масштабі до поточного покоління мереж додаються нові нейронні структури. Ми називаємо це «структурним дослідженням». У меншому масштабі поточні структури оптимізуються шляхом зміни їх параметрів («структурне використання»).

Структурна мутація в EANT додає або усуває прямий або зворотній зв'язок між нейронами або додає нову підмережу до лінійного геному. Структурна мутація не видаляє підмережу, оскільки видалення підмережі призводить до видалення всіх зв'язків, які надходять або виходять з підмережі. Це спричиняє величезну втрату продуктивності нейронної мережі. Структурна мутація діє тільки на вузлах нейронів. Припускаючи, що в популяції наразі N нейронів, кількість вузлів нейронів, структурна властивість яких буде змінена структурною мутацією, визначається як , де – кількість вузлів нейронів, структурна властивість яких змінюється внаслідок структурної мутації, а – ймовірність виконання структурної мутації [17]. При застосуванні структурної мутації кожен нейронний вузол перевіряється, чи буде він мутований або ні, шляхом вилучення випадкового числа з рівномірного розподілу між 0 та 1. Якщо поточне випадкове число менше , вузол нейрона буде мутований. Як тільки стане відомо, що вузол нейрона буде мутований, випадкове число знову витягується з рівномірного розподілу між 0 і 1 для визначення типу структурної мутації, яку потрібно виконати. Додавання з'єднань, додавання підмереж і видалення з'єднань мають однакові ймовірності

виконання. Незважаючи на те, що для лінійного геному відносно легко визначити оператор кросинговеру, він не використовується як оператор структурного пошуку, оскільки він призводить до іншої структури, яка може бути досягнута шляхом структурної мутації.

Як було сказано, алгоритм починається з мереж мінімальних структур і ускладнює їх на шляху еволюції. Дослідження нових структур розпочинаються, коли неможливо продовжити використання існуючих. Під використанням ми маємо на увазі оптимізацію ваг нейронних мереж. Для того, щоб ініціювати структурну мутацію, необхідно виявити умову неможливості подальшого використання існуючих структур. Для цієї мети використовується буфер довжиною для збереження найкращих значень пристосованості за останні поколінь. Якщо припустити, що ми знаходимося в -му поколінні, і якщо , де і є найкращими значеннями пристосованості -го та -го поколінь, менше за деякий поріг, то це означає, що система може не використовувати надалі існуючі структури і автоматично ініціюється структурна мутація для пошуку нових структур (ibid.). Іншими словами, якщо швидкість збільшення показника придатності найкращої особини, де швидкість визначається протягом поколінь, менша за деякий поріг, то ініціюється структурна мутація.

Імовірність виконання структурної мутації коригується таким чином, щоб щоразу, коли ініціюється структурна мутація, кількість нейронів, структурна властивість яких буде змінена, залишалася постійною. Імовірність мутації коригується за допомогою

. (2)

і – початкова і поточна кількість нейронів у популяції відповідно. Аналогічно, і є початковою і поточною ймовірністю структурної мутації відповідно. Таке пристосування призводить до сильної структурної мутації на початку еволюції. У міру еволюції мережі стають складнішими, а ефект структурної мутації зменшується. Тобто ефект структурної мутації зменшується у міру збільшення продуктивності нейронних мереж або в

міру наближення нейронних мереж до оптимальної мінімальної структури для даного початкового завдання.

Щоразу, коли відкриваються нові структури, вони тримаються протягом певної кількості поколінь незалежно від результатів оператора відбору. Це дасть їм час оптимізувати свої нещодавно придбані структури, перш ніж вони конкурують з іншими особами у всій популяції. Таким чином можна зберегти нові структурні відкриття еволюції до того, як вони зникнуть набагато раніше [13].

Використання існуючих структур починається з кластеризації структур відповідно до їх структурної схожості. Члени кластера робляться ідентичними щодо ваги вузлів після кожного покоління. Для оптимізації ваги структури використовується лише представник кластера. Кількість оцінок представника кластера на покоління визначається як

, (3)

де — кількість оцінок, використаних для оптимізації ваг даної структури, — середнє значення придатності індивідів у -му кластері, а — сума середніх значень придатності всіх кластерів у популяції, — чисельність популяції. Загальна схема алгоритму зображена на рисунку 2.3.

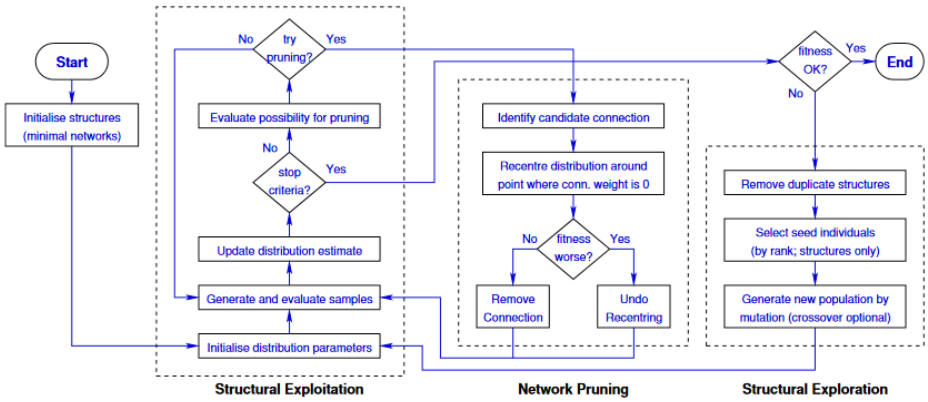


Рисунок 2.3 – Алгоритм EANT з обрізанням

CMA-ES використовується для оптимізації ваг нейронних мереж. Кожен кластер має власну коваріаційну матрицю і глобальний розмір

кроку . Поки кластер існує в популяції, коваріаційна матриця і глобальний розмір кроку продовжують розвиватися за допомогою адаптації коваріаційної матриці (CMA). Кількість оцінок на покоління, використаних для оптимізації ваг представника кластера, визначається рівнянням (3). Новоутворений кластер спочатку ініціалізує свою коваріаційну матрицю до одиничної матриці, а потім копіює записи коваріаційної матриці батьківського кластера, щоб підтримувалися вже розроблені кореляції між старими вузлами [9]. Однак глобальний розмір кроку ініціалізується деяким початковим значенням .

В CMA-ES алгоритмі використовують два основних принципів для адаптації параметрів розподілу пошуку. По-перше, принцип максимальної правдоподібності,що заснована на ідеї, підвищення ймовірності успішного вирішення кандидатів і пошуку кроків. Середній розподіл оновлюється, та ймовірність вибрати минулі успішні рішення є максимальним. Коваріаційна матриця розподілу оновлюється (поступово), що збільшує ймовірність того що будуть повторені попередні успішні кроки пошуку. Обидва оновлення можна розглядати як природний градієнт спуску. Крім того, CMA проводить повторний аналіз головних компонентів успішного кроку пошуку, зберігаючи при цьому всі головні осі.

Оцінка розподілу алгоритмів і методу крос-ентропії ґрунтується на дуже схожих ідеях, але оцінка коваріаційної матриці на максимальній ймовірності успішного вирішення вказується замість успішного пошуку кроку. По-друге, існують два шляхи еволюції розподілу - пошук і розвиток шляхів. Ці шляхи містять важливу інформацію про кореляцію між послідовними кроками. Зокрема, якщо послідовні кроки йдуть в тому ж напрямку, еволюція шляхів стають довгою. Еволюція шляху експлуатується в двох напрямках. Один шлях використовується для адаптації процедури коваріаційної матриці замість одного успішного кроку пошуку і полегшує набагато швидше, дисперсією збільшення сприятливих напрямків [8]. Інший шлях використовується для проведення додаткових

змін розміру кроку управління. Розміру кроку управління прагне зробити послідовні рухів розподілу. Розмір кроку контролю ефективно запобігає передчасному зближенню та призводить до оптимального розв'язку. Концепція CMA-ES зображено на рисунку 2.4. У поколіннях, розподіл форм може адаптуватися до еліпсоїдальної або форми хребта.

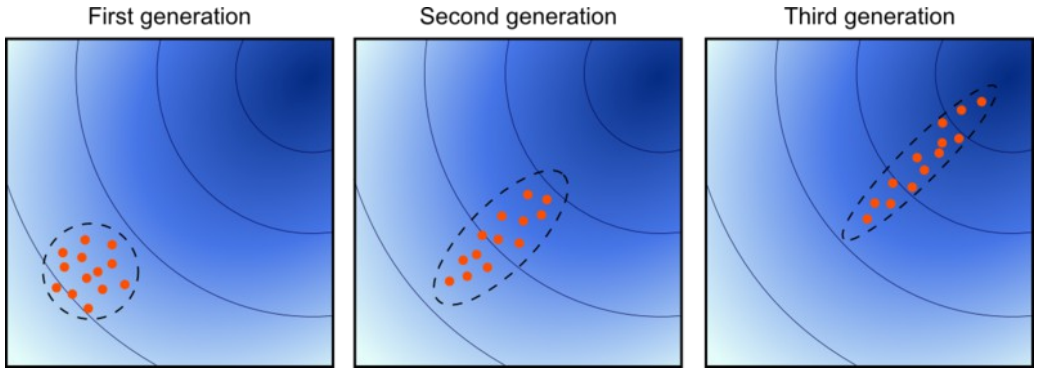


Рисунок 2.4 – Концепція адаптації коваріаційної матриці

2.3 Рекурентні нейронні мережі

Рекурентні нейронні мережі (RNN) це клас штучних нейронних мереж, у якому зв’язки між вузлами утворюють орієнтований або неорієнтований граф вздовж тимчасової послідовності. Це дозволяє йому демонструвати тимчасову динамічну поведінку. Похідні від нейронних мереж із прямим зв'язком, RNN можуть використовувати свій внутрішній стан (пам'ять) для обробки послідовностей змінної довжини вхідних даних. Це робить їх застосовними до таких завдань, як несегментоване, зв’язане розпізнавання рукописного введення або розпізнавання мовлення.

Для постановок керованого навчання з дискретним часом тренувальні послідовності входових векторів стають послідовностями активацій входових вузлів, по одному вектору на кожен момент часу. В кожен заданий момент часу кожен не входовий вузол обчислює свою поточну активацію як нелінійну функцію від зваженої суми активацій всіх вузлів, від яких до нього надходять з'єднання. Для деяких із виходових вузлів на певних тактах можуть бути задані вчителем цільові активації. Наприклад, якщо входова послідовність є мовленнєвим сигналом, що відповідає вимовленій цифрі, то кінцевий цільовий вихід у кінці послідовності може бути міткою, яка класифікує цю цифру. Для кожної послідовності її похибка є сумою відхилень усіх цільових сигналів від відповідних активацій, обчислених мережею. Для тренувального набору численних послідовностей загальна похибка є сумою похибок усіх окремих послідовностей.

У постановках навчання з підкріпленням не існує вчителя, який надавав би цільові сигнали для РНМ, натомість час від часу застосовується функція допасованості або функція винагороди для оцінювання продуктивності РНМ, яка впливає на її входовий потік через виходові вузли, з'єднані з приводами, що впливають на середовище [16].

Одним з популярних видів RNN є нейронна мережа Елмана. Ця

мережа отримується з багатошарового перцептрона введенням зворотних зв'язків, тільки зв'язки йдуть не від виходу мережі, а від виходів внутрішніх нейронів. Це дозволяє врахувати передісторію процесів, що спостерігаються і накопичувати інформацію для вироблення правильної стратегії управління. Ці мережі можуть застосовуються в системах управління рухомими об'єктами, тому що їх головною особливістю є запам'ятовування послідовностей. Використовується тришарова мережа (впорядкована на ілюстрації по горизонталі як x, y та z) з додаванням набору «контекстних вузлів». Існують з'єднання з середнього (прихованого) шару з цими контекстними вузлами з незмінними одиничними вагами. На кожному такті вхід поширюється стандартним прямим чином, а потім застосовується правило навчання. Незмінні зворотні з'єднання призводять до того, що контекстні вузли завжди зберігають копію попередніх значень прихованих вузлів (оскільки вони поширюються з'єднаннями до застосування правила навчання). Таким чином, мережа Елмана може зберігати свого роду стан, що дозволяє їй виконувати такі задачі, як передбачення послідовностей, що є за межами можливостей стандартного багатошарового перцептрону. Через цю особливість даний тип мереж показує дуже гарні результати у завданнях, пов’язаних з часовими рядами. Оскільки прогнозування погоди на основі попередніх даних і є прикладом таких рядів, використання такої є виправданим кроком для отримання гарних результатів у порівнянні з іншими методами і типами ШНМ. Приклад такої ШНМ зображено на рисунку 2.5.

Тренування ваг у таких нейронних мережаї можливо моделювати як нелінійну задачу глобальної оптимізації. Цільову функцію для оцінки допасованості або похибки певного вагового вектора може бути сформовано наступним чином. Спершу ваги в мережі встановлюються відповідно до цього вагового вектора. Далі, мережа оцінюється за тренувальною послідовністю. Як правило, для представлення похибки

поточного вагового вектора використовують суму квадратів різниць між передбаченнями та цільовими значеннями, вказаними в тренувальній послідовності. Потім для мінімізації цієї цільової функції може бути застосовано довільні методики глобальної оптимізації. Найуживанішим методом глобальної оптимізації для тренування РНМ є генетичні алгоритми, особливо в неструктурованих мережах. Спочатку генетичний алгоритм кодується вагами нейронної мережі в наперед визначеному порядку, коли один ген у хромосомі представляє одне зважене з'єднання, і так далі; вся мережа представляється єдиною хромосомою. Функція допасованості обчислюється наступним чином: 1) кожна вага, закодована в хромосомі, призначається відповідному зваженому з'єднанню мережі; 2) потім тренувальний набір зразків представляється мережі, яка поширює вхідні сигнали далі; 3) до функції допасованості повертається середньоквадратична похибка; 4) ця функція потім веде процес генетичного відбору [16].

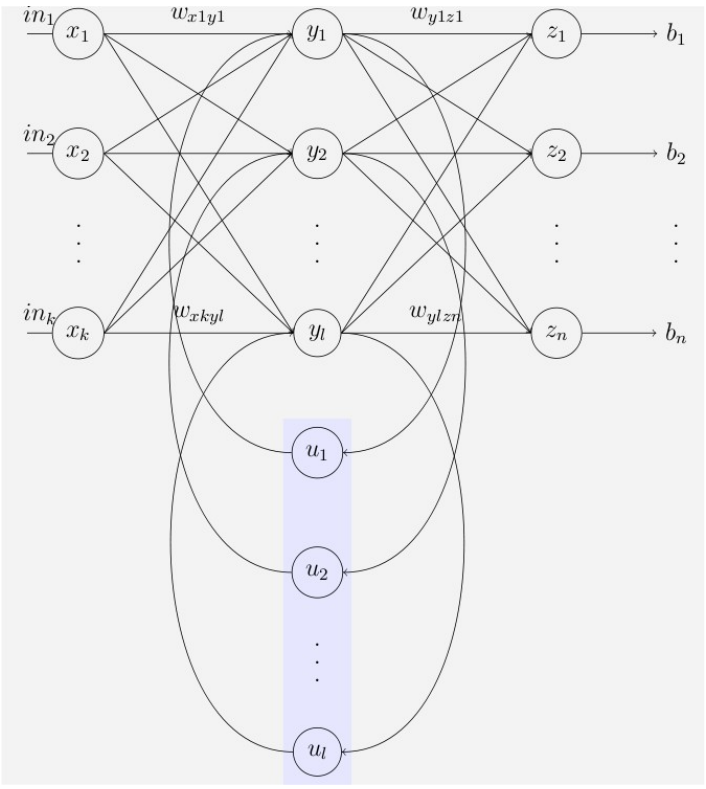


Рисунок 2.5 – Мережа Елмана

Популяцію складають багато хромосом; таким чином, багато різних нейронних мереж еволюціонують, поки не буде досягнуто критерію зупинки. Поширеною схемою зупинки є: 1) коли нейронна мережа засвоїла певний відсоток тренувальних даних, або 2) коли досягнуто мінімального значення середньоквадратичної похибки, або 3) коли було досягнуто максимального числа тренувальних поколінь. Критерій зупинки оцінюється функцією допасованості при отриманні нею оберненого значення середньоквадратичної похибки з кожної з нейронних мереж під час тренування. Отже, метою генетичного алгоритму є максимізувати функцію допасованості, знизивши таким чином середньоквадратичну похибку. Для пошуку доброго набору ваг можуть застосовуватися й інші методики глобальної (та/або еволюційної) оптимізації, такі як імітація відпалу та метод рою часток [2].

3 АНАЛІЗ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ

3.1 Фреймворк Keras

Keras це бібліотека програмного забезпечення з відкритим вихідним кодом, яка надає інтерфейс Python для штучних нейронних мереж. Keras виступає в якості інтерфейсу для бібліотеки TensorFlow. Створений для швидкого експериментування з глибокими нейронними мережами, він зосереджений на тому, щоб бути зручним, модульним і розширюваним.

Keras містить множину реалізацій часто використовуваних будівельних блоків нейронної мережі, таких як шари, цілі, функції активації, оптимізатори та множину інструментів, які полегшують роботу з графічними та текстовими даними для спрощення кодування, необхідного для написання коду глибоких нейронних мережі. На додаток до стандартних нейронних мереж, Keras підтримує згорткові та рекурентні нейронні мережі. Мається підтримка інших загальних методів, такі як dropout, batch normalization і pooling.

Keras є високорівневим шаром абстракціїї, що дозволяє легко створювати архітектуру бажаної нейронної мережі. У прикладі 3.1 наведено код для створення декількох слоїв мережі у вигляді послідовності (Sequental).

```
from keras.layers import Sequential
model = Sequential()
model.add(Dense(512, input_shape=(max_words,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```

Приклад 3.1 – Створення архітектури нейронної мережі

В наведеному прикладі ми маємо змогу легко створити наступні шари:

dense. Простий «щільний» шар, де усі нейрони з'єднані з попередніми;

activation. Функція активації, яку ми задаємо у вигляді випрямлювача;

dropout. Шар, який випадковим чином відкидує частину вхідної інформації з попереднього слою (розмір відкиду ми має змогу обирати самі).

Як можна побачити, заданими командами ми створили просту нейронну мережу всього за декілька рядків. Додатково, для більш зручного сприйняття, фреймворк надає нам інструменти для візуалізації створеної архітектури (приклад 3.2).

```
    from keras.utils import plot_model
    plot_model(model,              to_file='model.png',
show_shapes=True)
```

Приклад 3.2 – Візуалізація нейронної мережі

```
    model.compile(loss='categorical_crossentropy',
optimizer='adam',
metrics=['accuracy'])
```

Приклад 3.3 – Підготовка моделі

Отже, ми сформували нашу модель. Тепер потрібно підготувати її до роботи. Функція compile має деякі основні параметри. loss — це функція помилки, в нашому випадку це перехресна ентропія; optimizer — використовуваний оптимізатор, тут міг би бути звичайний стохастичний градієнтний спуск, але Adam показує кращу збіжність на задачі прогнозування; metrics — метрики, за яким вважається якість моделі, в нашому випадку — це точність (accuracy), тобто частка вірно вгаданих

відповідей.

Врешті, нам потрібно навчити нашу модель на заданих параметрах. Метод fit виконує навчання. Він приймає на вхід навчальну вибірку разом з мітками — x_train і y_train, розміром пакету batch_size, який обмежує кількість прикладів, поданих за раз, кількістю епох для навчання epochs (одна епоха — це один раз повністю пройдена моделлю навчальна вибірка), а також тим, яку частку навчальної вибірки віддати під валідацію validation_split. Повертає цей метод history — це історія помилок на кожному кроці навчання. І нарешті, тестування. Метод evaluate отримує на вхід тестову вибірку разом з мітками для неї. Метрика була задана ще при підготовці до роботи, так що більше нічого не потрібно. Приклад 3.4 ілюструє цей процес у програмному коді.

```
    history = model.fit(x_train, y_train,
batch_size=batch_size,
epochs=epochs,
verbose=1,
validation_split=0.1)
score = model.evaluate(x_test, y_test,
batch_size=batch_size)
```

Приклад 3.4 – Навчання і тестування моделі

4.2 Бібліотека EANT

Для EANT існує біблиотека, що дозволяє проводити нейроеволюцію, форму машинного навчання, яка тренує нейронні мережі з генетичним алгоритмом. Це засновано на моделі алгоритму EANT, вдосконаленому методі розвитку нейронних мереж шляхом комплексифікації. Нейронні мережі в EANT починають еволюцію з дуже простих геномів, які ростуть протягом наступних поколінь. Особи в популяції, що розвивається, групуються за подібністю за характеристиками, і кожна з них може

конкурувати лише з особинами одного виду (так званий інбридинг).

Бібліотека представляє собою абстракцію штучних нейронних мереж і генетичного програмування, надаючи найбільш зручні можливості до створення нейронних мереж та навчання і зміни архітектури за допомогою алгоритму, використовуючи невелику кількість програмного коду.

У поточній реалізації EANT на мові Python підтримується популяція окремих геномів. Кожен геном містить два набори генів, які описують, як побудувати штучну нейронну мережу:

вузлові гени, кожен з яких визначає окремий нейрон.

гени зв'язку, кожен з яких визначає єдиний зв'язок між нейронами (прямий або зворотній).

Щоб розробити рішення для заданої проблеми, користувач повинен надати функцію придатності, яка обчислює єдине дійсне число, яке вказує якість окремого геному: краща здатність вирішити проблему означає більш високий бал.

Операції розмноження і мутації можуть додавати вузли та/або з'єднання до геному. Так як алгоритм продовжує працювати, геноми (і нейронні мережі, які вони створюють) можуть ставати все більш і більш складними. Коли задане число поколінь досягнуто або коли принаймні одна особина (для функції критерію придатності max; інші налаштовуються) перевищує заданий користувачем поріг придатності, алгоритм завершується.

В процесі еволюції геному не відбувається процедура кросинговеру. Як вказано в описанні алгоритма, завдяки інбридингу і великої кількості мутацій, усі наявні і можливі структури з часом переберуть усі варіанти. Це дозволяє розришити простір можливих рішень на відміну від інших алгоритмів, хоча і коштує дещо більших потужностей для обчислення.

Після того, як реалізована функція пристосованості, потрібен додатковий шаблонний код, який виконує наступні кроки:

створює об'єкт eant.config.Config конфігурації з файлу конфігурації;

створює об’єкт eant.population.Population для заповнення за допомогою об’єкта конфігурації, створеного вище;

викликає метод run для об’єкта Population, надавши йому функцію придатності і (необов’язково) максимальну кількість поколінь, які ми хочемо запустити.

```
    def eval_genomes(genomes, config):
        for genome_id, genome in genomes:
            genome.fitness = 4.0
                                                net     =
eant.nn.FeedForwardNetwork.create(genome, config)
                    for  xi,  xo  in  zip(xor_inputs,
xor_outputs):
                output = net.activate(xi)
            genome.fitness -= (output[0] - xo[0])** 2
```

Приклад 3.5 – Створення функції обчислення нових геномів

Після повернення виклику методу run об’єкта population можна запросити елемент статистики population (eant.statistics.StatisticsReporter), щоб отримати кращий геном(и), що був під час запуску (приклад 3.7). У цьому прикладі ми беремо геном «переможця», який повертається методом pop.statistics.best_genome(). Додатково, у модулі візуалізації доступні функції для побудови графіків найкращої і середньої придатності порівняно з поколінням, побудови графіка зміни видів у порівнянні з поколінням і відображення структури мережі, описуваної геномом (приклад 3.6).

```
    node_names = {-1:’A’, -2: ‘B’, 0:’A XOR B’}
            visualize.draw_net(config,  winner,  True,
node_names=node_names)
             visualize.plot_stats(stats,  ylog=False,
view=True)
        visualize.plot_species(stats, view=True)
```

Приклад 3.6 – Візуалізація результатів

```
    config      =      eant.Config(eant.DefaultGenome,
eant.DefaultReproduction,
```

```
eant.DefaultSpeciesSet, eant.DefaultStagnation,
config_file)

         # Create the population, which is the top-
level object for a EANT run.
        p = eant.Population(config)

         # Add a stdout reporter to show progress in
the terminal.
        p.add_reporter(eant.StdOutReporter(True))
        stats = eant.StatisticsReporter()
        p.add_reporter(stats)
        p.add_reporter(eant.Checkpointer(5))
        # Run for up to 300 generations.
        winner = p.run(eval_genomes, 300)
        # Display the winning genome.
        print('\nBest genome:\n{!s}'.format(winner))
         # Show output of the most fit genome against
training data.
        print('\nOutput:')
                                    winner_net       =
eant.nn.FeedForwardNetwork.create(winner, config)
        for xi, xo in zip(xor_inputs, xor_outputs):
            output = winner_net.activate(xi)
         print("input {!r}, expected output {!r}, got
{!r}".format(xi, xo, output))
    p  =  eant.Checkpointer.restore_checkpoint('eant-
checkpoint-4')
        p.run(eval_genomes, 10)
```

Приклад 3.7 – Створення основних об'єктів для запуску EANT і показу результатів

4 ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ

5.1 Створення і налаштування моделі

Для навчання ШНМ обрано датасет глобальних температур земної поверхні і повітря Чиказького університету. Ці дані містять записи в різних частинах світу (міста, села, пустелі, водоймища, ліса тощо) за останні 50 років. У цьому датасеті наявні наступні дані:

довгота;

широта;

день року;

початкова температура повітря;

початкова температура поверхні;

рівень сонячної радіації;

альбедо;

прозорість атмосфери.

Усі ці значення є входом для ШНМ, які потрібно встановити явно. Також встановлюється тип функції активації виходів та прихованих нейронів. Приховані нейрони необов’язкові. Після створення геному ми створюємо популяцію так, як показано у прикладі 5.3.

```
import EANT
params = EANT.Parameters()
params.PopulationSize = 100
```

Приклад 4.1 – Ініціалізація

```
    genome = EANT.Genome(0, 3, 0, 2, False,
EANT.ActivationFunction.UNSIGNED_SIGMOID,
EANT.ActivationFunction.UNSIGNED_SIGMOID, 0, params,
0)
```

Приклад 4.2 – Створення геному

```
    pop = EANT.Population(genome, params, True, 1.0,
0) # the 0 is the RNG seed
```

Приклад 4.3 – Створення популяції

Останні два параметри визначають, чи слід розподіляти популяцію і скільки. Оскільки ми починаємо з нового геному, а не з того, який раніше було збережено, ми рандомізуємо початкову популяцію. Щоб почати просту еволюцію, нам потрібна функція оцінки пристосованості. Вона приймає геном як параметр і повертає число з плаваючею комою, яке відповідає фенотипу геному. Сам метод представлений у прикладі 4.4-4.5.

```
    def evaluate(genome):
        net = EANT.NeuralNetwork()
        genome.BuildPhenotype(net)
         # let's input just one pattern to the net,
activate it once and get the output
        net.Input( [ 1.0, 0.0, 1.0 ] )
        net.Activate()
```

Приклад 4.4 – Метод запуску еволюції

```
        output = net.Output()

         # the output can be used as any other Python
iterable. For the purposes of the tutorial,
           # we will consider the fitness of the
individual to be the neural network that outputs
constantly
          # 0.0 from the first output (the second
output is ignored)

        fitness = 1.0 - output[0]
        return fitness
```

Приклад 4.5 – Метод запуску еволюції (продовження)

Отже, оскільки ми вже маємо свою функцію оцінки, ми можемо ввести основний цикл еволюції поколінь, як показано у прикладі 4.6.

```
    for generation in range(100): # run for 100
generations
        genome_list = EANT.GetGenomeList(pop)
           # apply the evaluation function to all
genomes
        for genome in genome_list:
            fitness = evaluate(genome)
            genome.SetFitness(fitness)

        # advance to the next generation
        pop.Epoch()
```

Приклад 4.6 – Цикл запуску і відбору поколінь

Всередині викликаємих методів обчислюються нові структури за послідовністю дій, зображених на рисунку 4.1. Виділяються наступні головні етапі: ініціалізація, структурна есплуатація, відбір, оцінка пристосованості і структурне дослідження.

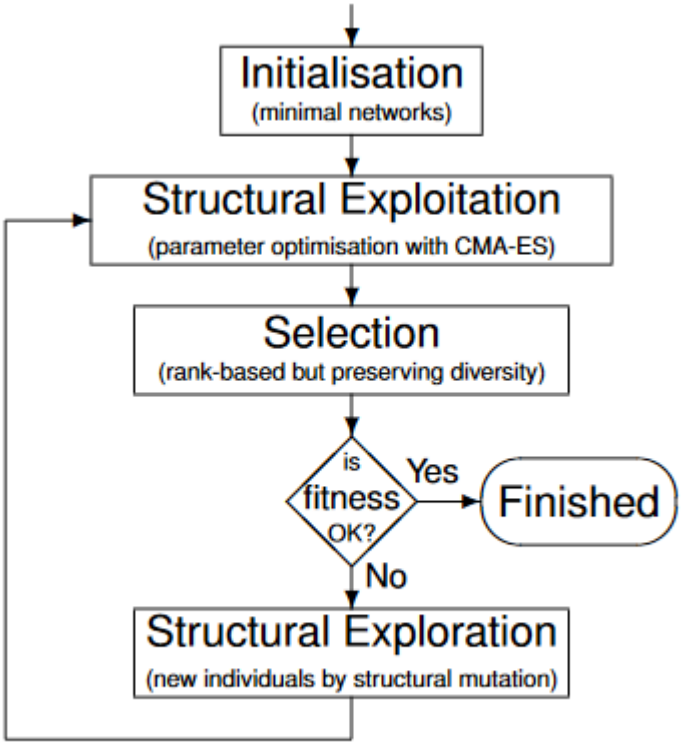


Рисунок 4.1 – Алгоритм EANT

Починаючи з найменшої структури, алгоритм створює початкову популяцію (у нашому випадку 100 осіб), оцінює їх пристосованість, оптимізує вагові коефіцієнти, обирає найпридатніших і застосовує оператори мутації. Таким чином, ми отримаємо деяку можливо оптимізовану структуру і маємо змогу перевірити її на обраних нами прикладах у порівнянні з іншими методами. Приклад деякого передбачення наведено на рисунку 4.2. На ньому мається дані про обраний часовий проміжок: температура поверхні, зміни у температурі, значення теплового потоку, значення прихованого теплового потоку, опускання і

підіймання радіації. Повний програмний код для вищеописаних етапів алгоритму наведено у додатку А.

```
>> python parametricscheme.py --latitude 47.6928 --longitude -122.3038
--day_of_year 229 --ground_temp 12 --surface_temp 22 --percent_net_radi
ation 0.2 --degrees C -fm 60
LSTM:     0
B:        2.5459587281658558
EOT:      -3.7759142377677044
TC:       -492.9911142377677
LST:      3.7834814293705374
HRA:      -123.24777855944194
delta:    13.38488935766939
zenith:   -0.18781454532142916
Q_S:      0
Q_Ld:     280.75967119838845
Q_Lu:     337.37979656244994
N_R:      -56.620125364061494
Q_H:      -11.324025072812299
Q_E:      -12.582250080902554
Q_G:      110.0
d_T_s:    -0.06116307866157713
T_s:      21.938836921338407
```

Рисунок 4.2 – Прогнозування температури поверхні

4.2 Навчання і візуалізація результатів

У результаті дослідів, імітаційна модель навчалася приблизно 500 поколінь з популяцією 100. Це дозволило за досить невеликий час отримати гарний результат апроксимації. Рівень мутації сягав значення 20%. Отримана структура містить лише 3 приховані шари. Після 400 поколінь значення пристосованості більше не змінювалося значно і постійно зростала кількість мутацій, намагаючись поширити простір пошуку, однак зупинка відбулася на встановленому ліміті. Згідно міркувань, подальша еволюція не призвела би до значних покращень моделі ШНМ.

На рисунках 4.3-4.4 представлено графіки залежності функції пристосованості і середньої помилки від кількості поколінь (чим менше, тим краще).

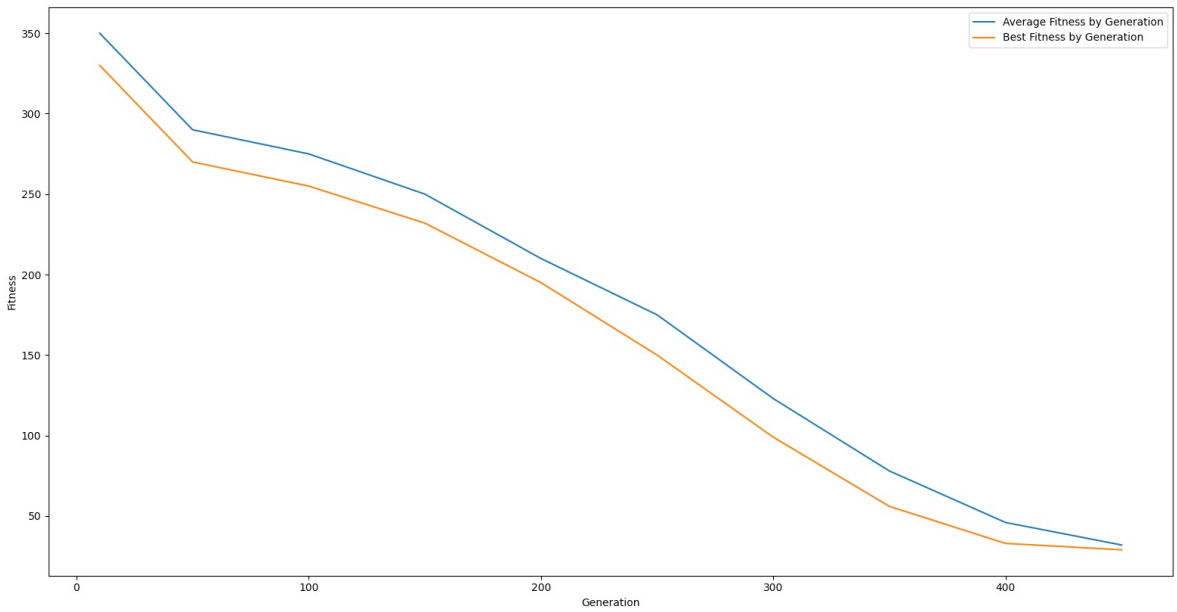


Рисунок 4.3 – Залежність між значення пристосованості і кількістю поколінь

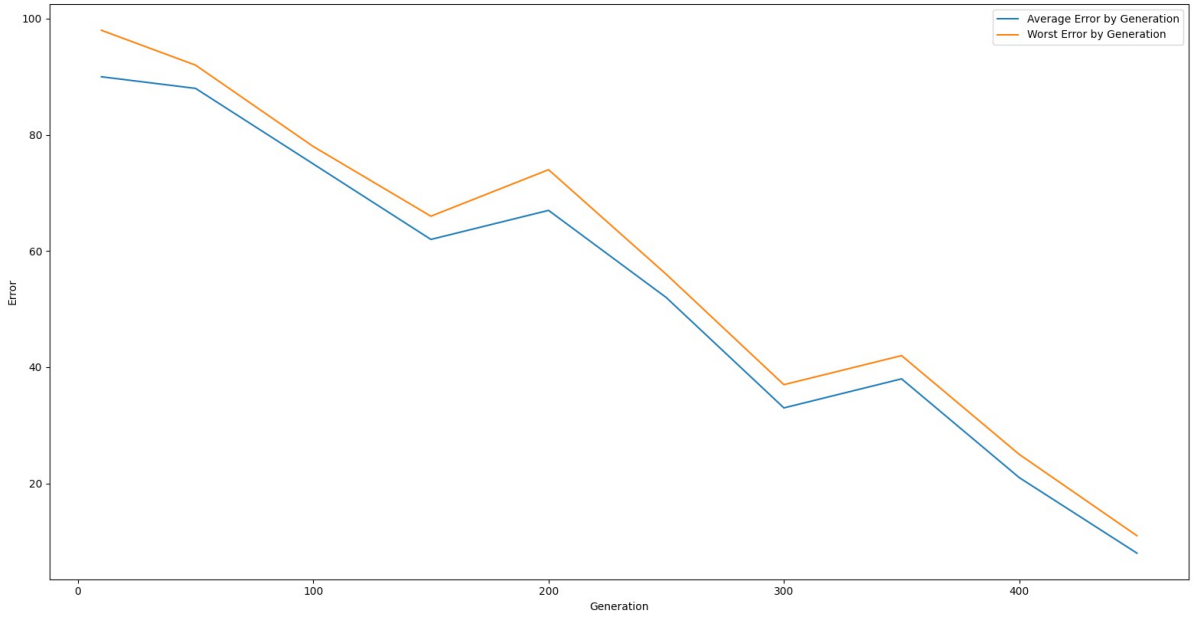


Рисунок 4.4 – Залежність між середньою помилкою і кількістю поколінь

5.3 Порівняння результатів

У ході випробувань також були порівняні різні ітеративні і метаевристичні алгоритми та їх придатність до вирішення поставленої задачі. Результати наведені у таблиці 4.1. Чим значення менше, тим краще

Оскільки обраний алгоритм є стохастичним за своєю природою, то для порівняння було обрано три інших подібних, а саме генетичний алгоритм ES-HyperNEAT і стохастичні мурашиний і бджолиний. Усі вони порівнювалися за однаковою кількістю еволюційованих поколінь. Порівняння зосереджено на значенні пристосованості у різні епохи і середньою помилку. Результати показують, що EANT дещо кращий при вирішенні поставленої задачі, хоча інші алгоритми також показують потенціал у використання в подібних сферах.

Таблиця 4.1 – Порівняння різних алгоритмів для задачі комівояжера

Метод	Середнє значення пристосованості (100 епох)	Середнє значення пристосованості (200 епох)	Середня помилка (300 епох)
EANT	275	210	38
Мурашиний алгоритм	269	223	43
Бджолиний алгоритм	280	230	45
ES-HyperNEAT	250	215	40

ВИСНОВКИ

У даній роботі були проаналізовані задача прогнозування температури земної поверхні та існуючі методи її розв’язання, запропоновано використовувати для цієї мети штучну нейронну мережу, досліджено предметну галузь штучних нейронних мереж і метод їх послідовного самонавчання.

Були проаналізовані архітектури простих штучних нейронних мереж, можливі функції активації, кількість нейронів у кожному слою, параметри налаштування мережі, кількість слоїв та найбільш відповідний набір параметрів для заданої задачі.

Для навчання з підкріпленням ШНМ був обраний генетичний алгоритм. Докладно розглянуто функціонування генетичного алгоритму, його основні етапи, переваги та недоліки, а також його взаємодія зі штучною нейронною мережею під час навчання.

За допомогою засобів фреймворка Keras побудовано штучну нейронну мережу для прогнозування температури земної поверхні. Проведено навчання ШНМ на датасет глобальних температур земної поверхні і повітря Чиказького університету з використанням генетичеого алгоритма EANT .

Порівняння результатів альтернативних методів рішення задачі прогнозування температури з результатами, отриманими ШНМ, що була сконструйована за допомогою використаного варіанта генетичного алгоритму, продемонструвало переваги останньої у критеріях ефективності, часу навчання і затрат обчислювальних потужностей.

Були виявлені певні проблеми проаналізованих алгоритмів, головні з яких є досить довге навчання при великих вхідних даних та труднощі з порівнянням результатів.

Метою роботи стало вирішення даних проблем і створення системи глибинної штучної нейронної мережі, а саме рекурентної ШНМ, та її

послідовного навчання, яка в процесі своєї роботи формує прогнозування температури відносно заданих чинників на обраній території. Надані графічні порівняння з іншими обчислювальними методами для прогнозування температури поверхні. За даними дослідженнями, описаний у роботі підхід продемонстрував результати прогнозування, кращі за отримані іншими алгоритмами.

ПЕРЕЛІК ПОСИЛАНЬ

2. F. J. Gomez and R. Miikkulainen. Robust non-linear control through neuroevolution. Technical report, Department of Computer Sciences, The University of Texas, Austin, TX 78712, U.S.A., 2002.

3. F. Pasemann. Evolving neurocontrollers for balancing an inverted pendulum. Network: Computation in Neural Systems, 9:495–511, 1998.

4. Kenneth O. Stanley; Bobby D. Bryant, Risto Miikkulainen. Evolving Adaptive Neural Networks with and without Adaptive Synapses. Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC-2003). 2003.

5. C. Igel. Neuroevolution for reinforcement learning using evolution strategies. In R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam, and T. Gedeon, editors, Congress on Evolutionary Computation (CEC2003), volume 4, pages 2588–2595. IEEE Press, 2003.

6. K. O. Stanley. Efficient Evolution of Neural Networks through Complexification. PhD thesis, Artificial Intelligence Laboratory. The University of Texas at Austin., Austin, TX 78712, U.S.A., August 2004.

7. P. J. Angeline, G. M. Saunders, and J. B. Pollack. An evolutionary algorithm that con- structs recurrent neural networks. IEEE Transactions on Neural Networks, 5:54–65, 1994.

8. X. Yao. Evolving artificial neural networks. Proceedings of the IEEE, 87(9):1423–1447, 1999.

9. N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation, 9(2):159–195, 2001.

10. Holland, John. Adaptation in Natural and Artificial Systems. Cambridge, MA: MIT Press. 1992. ISBN 978-0262581110.

11. J. S. Astor and C. Adami. A developmental model for the evolution of artificial neural networks. Artificial Life, 6(3):189–218, 2000.

12. F. Gruau, D. Whitley, and L. Pyeatt. A comparison between cellular

encoding and direct encoding for genetic neural networks. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, Genetic Programming: Proceedings of the First Annual Conference, pages 81–89, Standford University, CA, USA, 1996. MIT Press.

13. Бодянский Е. В. Искусственные нейронные сети: архитектуры, обучение, применения. Харьков ТЕЛЕТЕХ, 2004. 372 с.

14. Kassahun and G. Sommer, “Efficient reinforcement learning through evolutionary acquisition of neural topologies,” in Proceedings of the 13th European Symposium on Artificial Neural Networks (ESANN 2005), Bruges, Belgium, April 2005, pp. 259–266.

15. C. Igel, “Neuroevolution for reinforcement learning using evolution strategies,” in Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2003). IEEE Press, 2003, pp. 2588–2595.

16. G. Cybenko, “Approximation by superposition of sigmoidal functions,”Mathematics of Control, Signals and Systems, vol. 2, no. 4, pp. 303–314, December 1989.

17. J. Clune, K. Stanley, R. Pennock, and C. Ofria. On the performance of indirect encoding across the continuum of regularity. Evolutionary Computation, IEEE Transactions on, 15(3): 346–367, 2011.

18. C. M. Bishop, Neural Networks for Pattern Recognition. Oxford, UK: Oxford University Press, 1995.

19. T. Kavzo ̆glu and P. M. Mather, “Assessing artificial neural network pruning algorithms,” in Proceedings of the 24th Annual Conference and Exhibition of the Remote Sensing Society (RSS 1998), Greenwich, UK, 1998, pp. 603–609.

20. Hancock, P. J. B. Genetic algorithms and permutation problems: A comparison of recombination operators for neural net structure specification. IEEE Computer Society Press, 1992.