КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «магістр»

НА ТЕМУ:

Методи машинного навчання у прогнозуванні результатів спортивних змагань

Галузь знань: 12 «Інформаційні технології»
Спеціальність: 122 «Комп'ютерні науки»
Освітньо-наукова програма «Технології штучного інтелекту»

Виконав:
студент 2 курсу магістратури, групи ТШІ-21, Савченко Роман Вікторович

Науковий керівник: Сорока Петро Миколайович
кандидат фізико-математичних наук, доцент

Засвідчую, що в цій кваліфікаційній роботі немає запозичень з праць інших авторів без відповідних посилань

Студент _____
підпис

Кваліфікаційна робота допущена до захисту
рішенням кафедри *інтелектуальних технологій*

Протокол № ____ від «____» травня 2021 р.

Зав. кафедри _____ доц. Іларіонов О.Є.
підпис

Київ 2021

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ASP	Active Server Pages
CSS	Cascading Style Sheet
CSV	Comma Separated Values
GUI	Graphics User Interface
HTML	HyperText Markup Language
JSP	Java Server Pages
PC	Personal Computer
PHP	Personal Home Page, **P**HP **H**ypertext **P**reprocessor
SQL	Structured Queries Language
WWW	World Wide Web
XML	eXtensible Markup Language
БД	База даних
ВНЗ	Вищий навчальний заклад
ЗМІ	Засоби масової інформації
ІТ	Інформаційні технології
ОО	Об’єктно-орієнтований (-на, -не)
ООП	Об’єктно-орієнтоване програмування
ПЗ	Програмне забезпечення
ПК	Персональний комп’ютер
СУБД	Система управління базами даних
ТНМ	Теорія нечітких множин
ШНМ	Штучна нейронна мережа

ВСТУП

З моменту свого виникнення сучасні інформаційні технології значно розширили спектр галузей свого практичного застосування. Один із магістральних напрямків їх розвитку полягає у розробці методів та засобів прогнозування у найширшому сенсі.

Дійсно, у будь-якому процесі (що розглядається не у свій початковий момент) є певна передісторія та хід його виконання. Ця інформація завжди може бути представлена у цифровій формі (навіть при словесному описі у якісних категоріях дослідник може застосувати механізм теорії нечітких множин для формалізації проблеми «до числа») і за допомогою спеціальних методів, які в цілому можуть бути віднесені до галузі ІТ (або надзвичайно тісно пов’язані із нею, зважаючи на широке використання комп’ютерної техніки у будь-якому із них), у цифровій же формі можна отримати і прогнозні значення для певного процесу, тобто передбачити його хід у найближчому (зазвичай, із більшим ступенем достовірності) або і більш віддаленому майбутньому.

Прогнозування є чи не найпершою задачею, для якої взагалі створювалася комп’ютерна техніка (перші електронно-обчислювальні машини – ЕОМ – розроблялися переважно для розрахунку процесу польоту бойових ракет, який, при розгляді цього процесу як стохастичного – з урахуванням різноманітних випадкових збурень, також може бути віднесений до прогнозів). На сьогодні методи прогнозування активно використовуються в економіці, соціології, політиці, біології, метеорології, медицині, природничих науках тощо. Серед галузей застосування також є і спорт.

У спорті існує ціла низка причин, через які прогнозування результатів спортивних подій є актуальною і вкрай необхідною задачею, що потребує розроблення удосконалених (або нових) ефективних методів розв’язання. У першу чергу, реальна оцінка можливого результату події може підвищити

ефективність планування всього процесу підготовки, визначатиме тактику та стратегію під час змагання тощо. Наприклад, футбольні команди під час матчів дуже високого рівня, але коли результат змагання добре прогнозований (наприклад, коли ослаблена травмами команда виступає проти суперника того ж рівня, але такого, що знаходиться на піку), може дозволити собі ризиковані рішення: спробувати нову схему гри, дати спробувати нові амплуа для досвідчених гравців, випустити на поле новобранців, аби дати їм можливість «відчути серйозну гру», тощо. Так само і в тенісі або боксі: якщо прогноз поєдинку явно схиляється в одну із сторін, інша може запросити перенос на наступний період для створення можливості кращої підготовки. В усіх видах спорту наявність гарного прогнозу є важливим фактором, що може бути використаний для підвищення шансів учасників змагань на перемогу.

Однак, існує й інша не менш значима група осіб (або навіть і більш чисельна), для яких важливим є якісне прогнозування результату – це вболівальники, що роблять ставки в букмекерських конторах, на тоталізаторах, беруть участь у спортивних лотереях. Тут якість прогнозу прямим чином впливає на прибутковість подібних занять, тому використання ефективних методів також є надзвичайно важливим і значущим.

Зважаючи на наведені факти, розробка (удосконалення) ефективних методів прогнозування результатів спортивних змагань є актуальною науково-технічною задачею, яку доцільно розв’язувати, зокрема, в рамках написання магістерської роботи.

Метою роботи є підвищення якості прогнозування результатів спортивних змагань шляхом удосконалення однієї із існуючих методик, які вже себе зарекомендували.

Для досягнення мети потрібно розв’язати наступні *задачі дослідження*:

- проаналізувати сучасну науково-технічну літературу та електронні джерела на предмет сучасних ефективних способів прогнозування;

- обґрунтовано обрати один із ефективних способів, що має хороший потенціал для покращення;

- внести нові характеристики у процес прогнозування, удосконаливши обраний раніше алгоритм;

- вибрати технології та засоби розробки;

- здійснити програмну реалізацію удосконаленого алгоритму;

- виконати тестування та оцінити ефективність усієї роботи.

Об'єкт дослідження – прогнозування результатів спортивних подій.

Предмет дослідження – методика прогнозування результатів спортивних подій на основі методів машинного навчання.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМИ ПРОГНОЗУВАННЯ РЕЗУЛЬТАТІВ СПОРТИВНИХ ЗМАГАНЬ

1.1 Особливості процесу прогнозування результатів змагань за видами спорту

Незважаючи на спільну направленість процесу прогнозування результатів змагань за різними видами спорту, все ж усі вони мають власну специфіку, яку, відповідно, потрібно розглянути та перейти до певних їх видів.

Найбільш популярним видом спорту, очевидно, є футбол, що характеризується дихотомічним характером змагань, коли результат матчу визначається між двома супротивниками (якими є цілі команди гравців, однак, оскільки всі вони прикладають зусиль для досягнення спільного результату, команду можна вважати монолітним, єдиним супротивником). Результат матчу на найвищому рівні абстракції може приймати три різних варіанти: перемога першої команди (що еквівалентно програшу другої), нічия та перемога другої команди (що еквівалентно програшу першої). Більш детально результат визначається рахунком гри, який у переважній більшості випадків знаходиться у межах 3:0 – 0:3 та дуже рідко виходить за ці межі. Матч складається з двох таймів, рахунок після першого тайму зазвичай не вважається важливим результатом, а часто просто ігнорується.

Наступним дуже популярним видом спорту (другим по кількості його телеглядачів у світі) є перегони класу «Формула 1». У цьому випадку змагання відбувається між приблизно двадцятьма суперниками, які, навіть незважаючи на приналежність до однієї команди, повинні виступати виключно з метою власної перемоги, а командна тактика суворо заборонена. Таким чином, за умови дотримання правил, маємо близько 20 суперників, які у безперервній гонці, що відбувається близько двох годин, визначають переможців, що першими

доїжджають до фінішу, тобто проходять задану дистанцію гонки. Фінальний протокол, очевидно, відображає час проходження траси кожним учасником і чим меншим є цей час, тим кращий результат пілота. Відповідно, задачу у даному виді спорту можна звести до мінімізації часу виконання певної дії (завдання). Ранжування за числовим показником часто зустрічається і в інших індивідуальних видах спорту, наприклад легкій атлетиці (по мінімальному часу виконання, по максимальній дальності, по максимальній висоті тощо).

Розглянуті приклади відносяться до достатньо об’єктивно оцінюваних змагань, але існує чимала кількість таких змагань, де переможців визначає експертне журі, тобто результат є суб’єктивним (оцінка краси рухів, домінування на рингу, синхронності дій тощо). Через дуже слабку формалізацію процесу виставлення кінцевої оцінки, а отже і процесу її прогнозування, такі види спорту не розглядатимемо.

Нарешті, ще одним дуже популярним видом спорту, якому притаманні об’єктивні кінцеві результати, є теніс. Асоціація професіоналів тенісу (ATP) щороку проводить понад 60 професійних тенісних турнірів у 30 країнах світу, збираючи велику кількість глядачів. Матчі найвищого рівня неодноразово ставили рекорди кількості переглядів серед телевізійних трансляцій, сягаючи десятків мільйонів.

Зростання популярності цього виду спорту, що поєднується з розширенням ринку онлайн-ставок на спорт, призвело до значного збільшення обсягів ставок на теніс в останні роки. Загальна сума ставок на одну важливу зустріч тенісистів може перевищувати 100 мільйонів доларів.

Потенційний прибуток, а також науковий інтерес сприяє пошуку дієвих методів прогнозування тенісних матчів.

Варто зауважити, що система підрахунку балів у тенісі має ієрархічну структуру: так, матч складається із сетів, які у свою чергу складаються з геймів, які складаються з окремих очок («поінт»). Більшість сучасних підходів до

прогнозування тенісу використовують цю структуру для визначення ієрархічних виразів для ймовірності того, що гравець здобуде перемогу в матчі.

Припускаючи, що очки незалежно і однаково розподілені, вирази потребують лише ймовірності того, що два гравці виграють очко на власній подачі. З цієї основної статистики, яку легко обчислити з історичних даних, наявних у вільному доступі, стає можливо вивести ймовірність того, що гравець виграє гейм, потім – сет і, нарешті, матч.

Барнетт [1], O'Malley [2] і Knottenbelt [3] у своїх роботах визначили такі ієрархічні моделі для обчислення ймовірності виграшу очка на подачі, використовуючи лише матчі зі спільними супротивниками гравців замість усіх минулих супротивників. Це зменшує упередженість внаслідок відмінності рівня гри супротивників. Мадурська [4] вдосконалила модель, використавши різні ймовірності виграшу в різних сетах і дозволяючи моделі відображати те, як змінюється ефективність гравця в ході матчу.

Такий математичний підхід, хоч елегантний, не є ідеальним. Представляючи рівень гравців, використовуючи лише одне значення (виграні очки), метод не враховує надзвичайно великої кількості надзвичайно важливих параметрів. Наприклад, сприйнятливість гравця до певної ігрової стратегії опонента, час після останньої травми або накопичену втому від попереднього матча. Крім того, існують важливі характеристики самого матчу (місце проведення, погодні умови, покриття тощо). Враховуючи наявність величезної кількості різноманітних історичних даних про теніс, альтернативний підхід до прогнозування тенісу може базуватися на машинному навчанні.

Параметри гравців та особливості самого матчу в поєднанні з результатом матчу можуть складати набір даних для застосування машинного навчання. Алгоритми «навчання з учителем» можуть бути застосовані для визначення функції прогнозування результатів нових матчів.

Незважаючи на те, що машинне навчання є природним кандидатом у проблемі прогнозування тенісних матчів, йому приділялося менше уваги порівняно зі стохастичними ієрархічними підходами. Більшість минулих спроб використовували логістичну регресію. Наприклад, Кларк і Діт [5] використовують логістичну регресійну модель, враховуючи різниці в рейтингових балах ATP двох гравців, для прогнозування результату сету. А Классен та Магнус [6] показують, що очки не є ні незалежними, ні однаково розподіленими. Однак вони знаходять це відхилення незначним, і використання цього припущення часто дає хороші наближення.

Беручи до уваги наведену інформацію, можна сказати, що інтерес представляє використання результатів першого сету даного конкретного матчу для прогнозування результату другого, а потім і всієї гри.

1.2 Аналітичний огляд існуючих методів прогнозування результатів спортивних подій для різних видів спорту та для тенісу зокрема

Проблемі прогнозування результатів спортивних подій присвячено чимало наукових джерел, причому (як це не дивно) переважна частка із них спрямована не на допомогу спортсменам, а на особливості виконання ставок (що показує серйозне ставлення науковців до цього напрямку досліджень). Цікаво, що для вирішення цієї проблеми науковці залучають досить різні математичні підходи (іноді – діаметрально протилежні), причому усі вони декларують гарні результати прогнозування. Розглянемо деякі з них.

Укрупнено усі методи прогнозування, зокрема і результатів спортивних подій, можна поділити на:

- статистичні методи;
- популярні на сьогоднішній день методи з використанням штучного інтелекту та машинного навчання (зокрема, теорія нейронних мереж та апарат нечіткої логіки);

- інші методи інтелектуального аналізу даних, що використовують більш екзотичні (і малопоширені) алгоритми, ніж вищенаведені.

1.2.1 Методи математичної статистики

Значна частка наукових робіт у цій галузі присвячена традиційним статистичним методам з урахуванням кваліметричних показників спортивних команд (учасників змагань). Так, наприклад, в роботах [7-11] різними авторами розглядається практично один і той самий підхід, при якому послідовність дій для виконання прогнозу є наступною:

- обирається група показників, які, на думку авторів, є важливими для суті поставленої задачі;
- величина кожного показника нормалізується, наприклад, шляхом простого ділення на якесь реперне максимальне значення цього показника;
- для кожного показника обирається його «вага» для даної предметної галузі (очевидно, що для прогнозування різних видів спорту, ваги будуть різнитися, адже, наприклад, травмування одного члена команди може бути мало суттєвим у футболі, але критичним у бобслеї);
- обчислюється підсумкове, результуюче значення якогось інтегрального показника, на основі якого за певними правилами (зазвичай, це прості інтервальні нерівності, одній з яких і відповідає отриманий підсумковий результат розрахунків) робиться висновок про кінцевий результат спортивного змагання.

При такому підході не враховується інформація про параметри попередніх станів процесу, а тільки відомі обставини майбутнього стану. Моделі, що відповідають такому ходу дій можна назвати кваліметричними, скоринговими (від англ. score – очки, бали), зваженим сумуванням, адитивними, зважено-адитивними тощо.

1.2.2 Методи галузі штучного інтелекту

Окремим класом таких систем, які, в принципі, можуть застосовуватися для задач прогнозування, є розв'язки на базі нейронних мереж, яким присвячена чи не найбільша відносна частка усіх наукових робіт останніх років зі спортивного прогнозування, наприклад [12-16]. Така ситуація у великій мірі обумовлена певним «бумом», що спостерігається в галузі штучного інтелекту та суміжних напрямках навколо поняття нейронної мережі. Цей об'єкт широко застосовується для задач розпізнавання, класифікації, апроксимації, і зважаючи на це – і для прогнозування.

Штучною нейронною мережею (далі – ШНМ), як відомо, називають систему, яка, приймаючи на вхід вектор вхідних величин, за складними (нелінійними) внутрішніми законами виробляє вектор вихідних величин, що є відносно задовільним розв'язком поставленої задачі.

Елементарною складовою ШНМ є один нейрон (рис. 1.1). Це об'єкт, що має певну кількість входів (на рисунку – *n* входів), які називають дендритами та один вихід, що називається аксоном.

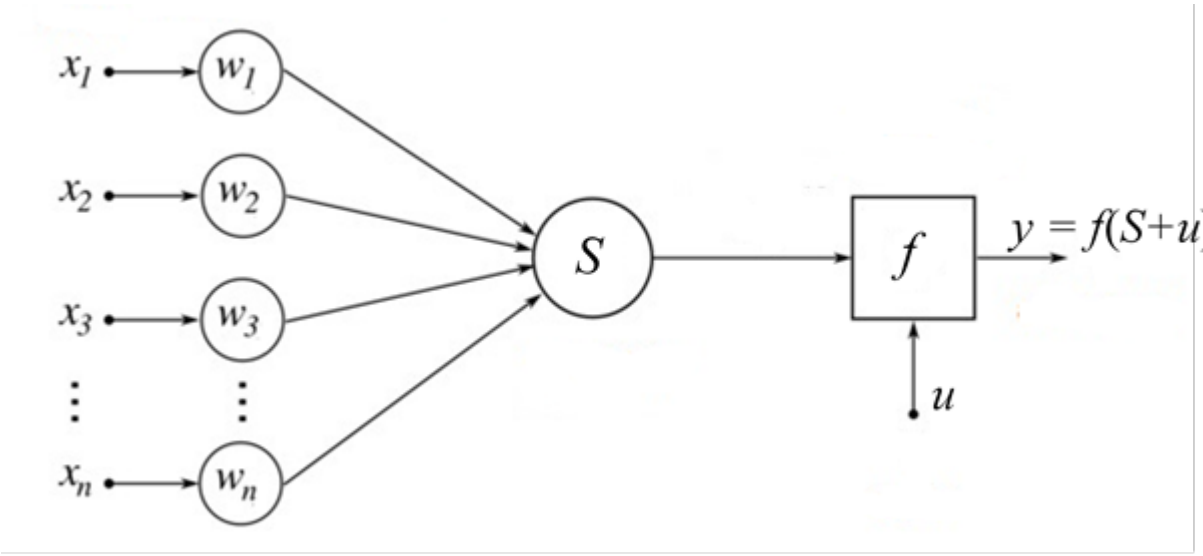


Рис. 1.1 Структура штучного нейрону

На входи нейрону подаються вхідні сигнали x_1, x_2, …, x_n, що є виходами попередніх нейронів, розташованих раніше по ходу проходження сигналу. Кожен вхідний сигнал x_i множиться на певне число w_i, що називають вагою (або синаптичною вагою) відповідного входу. Усі таким чином зважені вхідні сигнали подаються на суматор, який виробляє зважену суму S виду:

$$S = \sum_{i=1}^{n} w_i x_i \,. \qquad (1.1)$$

До цієї зваженої суми вхідних сигналів може додаватися зсув або зміщення u (у поширеному частинному випадку зміщення відсутнє, тобто $u = 0$) і від цієї суми виробляється певна функція f, що називається функцією активації; так і формується вихідний сигнал нейрону y, що передається далі по нейронній мережі:

$$y = f(S + u). \qquad (1.2)$$

Відповідно, комбінуючи (1.1) та (1.2), маємо функцію, що здійснює штучний нейрон:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + u \right). \qquad (1.3)$$

При аналізі такого способу завдання функції, що здійснює нейрон, бачимо, що важливу роль для кожного нейрона відіграють три речі (особливо перші дві):

- набір конкретних значень ваг w_i, що описують кожен вхід нейрона;

- вид функції активації f даного нейрона;

- наявність зсуву або порогу активації u (який може бути як додатним, так і від'ємним).

Зважаючи на те, що звичайна ШНМ складається як мінімум із кількох нейронів (а частіше – з десятків, чи навіть сотень нейронів), а в кожного з них є багато входів, то задача вибору вагових коефіцієнтів w_i стає дуже серйозною за своєю складністю. Власне, тривалий процес задання конкретних «правильних» значень цих вагових коефіцієнтів називають процесом навчання нейронної

мережі. Існують різні алгоритми навчання нейронних мереж, які укрупнено можна поділити на дві групи: «з учителем» та «без учителя».

Вид функції активації *f*, очевидно, також є дуже важливим фактором для функціонування нейронів, хоча й у значно меншій мірі, ніж вибір синаптичних вагових коефіцієнтів. На рис. 1.2 наведено найбільш поширені види функцій активації, що найчастіше використовуються на практиці при роботі з нейронними мережами. З точки зору обчислювальної складності кращим є використання порогових функцій, що фактично являють собою константну залежність (або лінійних, але не сигмоїдних).

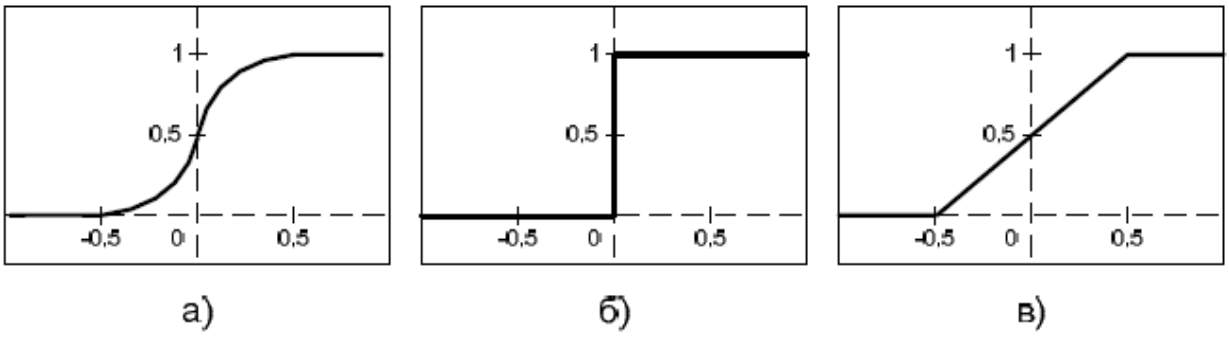


Рис. 1.2 Найбільш поширені типи функцій активації:
а – сигмоїдна, б – порогова, в – лінійна

При побудові нейронної мережі, крім характеристик окремих нейронів, важливим стає також спосіб їх з'єднання (так само, як і їх загальна кількість, організація нейронів у групи, шари тощо).

Найпростішим і в той же час найпоширенішим способом з'єднання нейронів у конкретну ШНМ є персептрон, що обумовлено широким спектром задач, які можна розв'язувати за допомогою ШНМ з такою внутрішньою структурою.

Беручи до уваги усі особливості ШНМ, а також усіх сформованих вище кроків по розв'язанню задачі прогнозування, бачимо, що з формальної точки зору нейронні мережі є ідеальним інструментом для розв'язання задачі

прогнозування. Відповідно, даний механізм візьмемо за основу при розробці власного методу прогнозування результатів спортивних змагань.

1.2.3 Апарат теорії нечітких множин

Через велику популярність доцільно розглянути також і апарат теорії нечітких множин, як засіб для прогнозування результатів змагань. Отже, одним із дуже популярних методів отримання кінцевого розв'язку в задачі прогнозування результатів спортивних змагань є застосування апарату нечіткої логіки (fuzzy logic), що є ще одним перспективним механізмом, який також широко застосовується в галузі інтелектуального аналізу даних (ІАД) [11-15]. Перевагою таких систем є можливість роботи із нечисловими, розмитими, або взагалі якісними, вираженими за допомогою текстового опису, лінгвістичними даними. Процедура фазифікації вхідної інформації, в якій використовуються лінгвістичні змінні дозволяє поставити у відповідність елементам множини $\vec{u}_{n+1}$ їх текстові описи, а потім використати простий та зрозумілий набір правил нечітких продукцій для вироблення кінцевого розв'язку у нечіткій формі. Після застосування процедури дефазифікації отримуємо конкретне значення прогнозованої величини. Головною перевагою при такому підході є використання простої бази правил, яку може розробити на основі аналізу логіки роботи системи (або ходу протікання процесу) людина з базовими знаннями предметної області, тобто відсутність необхідності зведення складної математичної моделі (можливо на основі методів багатовимірного регресійного аналізу й інших статистичних методів). Правила мають структуру на зразок «якщо кількість травм серед захисників є високою, то висока імовірність пропустити гол», «якщо нападаючі у гарній формі, то висока імовірність забити гол» тощо.

Зважаючи на широку поширеність систем нечіткого виведення, розглянемо докладніше процес їх використання (зокрема для задач прогнозування). Процес

полягає у циклічному виконанні процедури нечіткого виводу за допомогою системи нечіткого виводу. Єдиним кроком, що не виконується під час робочого часу системи, а реалізується лише один раз – при розробці системи – є створення системи нечітких правил продукцій (бази правил нечітких продукцій).

Під системою нечітких правил продукцій мається на увазі узгоджена множина окремих правил нечітких продукцій, побудованих на основі виразів виду:

ПРАВИЛО <1>: ЯКЩО «$\beta_1 \in \alpha_{11}$» ТО «$\beta_2 \in \alpha_{21}$», (1.4)

де «є» означає «являється», або англійською «is».

Тут нечіткі вирази антецеденту *A* та консеквенту *B* розписані у формі «$\beta_i \in \alpha_{ij}$» через лінгвістичні змінні β_i, деякі з яких є вхідними (це змінні, що беруть участь у антецедентах), деякі – вихідними (змінні, які входять до консеквентів). α_{ij} – певний терм із терм-множини лінгвістичної змінної β_i.

Таким чином, першим кроком процедури нечіткого виводу, який до того ж виконується лише один раз, є створення бази правил нечітких продукцій. Це задача, яку має виконувати експерт (експерти) у даній предметній галузі, оскільки від якості цієї системи (її повноти та адекватності кожного правила) залежить успішна робота системи прогнозування в майбутньому. Помилки при проектуванні бази правил можуть призводити до катастрофічних наслідків, причому найбільша частка складних проблем має за причину саме недоліки системи нечітких правил продукцій.

Важливою властивістю бази правил має бути її узгодженість, тобто правила не повинні суперечити одне одному (так завжди і є, коли розробку виконує один експерт, що має закінчений, усталений і ефективний погляд на прогнозування задач заданого класу). При розробці бази правил командою експертів, перевірка на узгодженість має виконуватися окремим підетапом.

Після зведення бази правил нечітких продукцій та її реалізації у певному електронному апаратному забезпеченні розпочинається процес нормального

функціонування системи прогнозування, що є ітеративним, циклічним і кожна ітерація складається з наступних складових (рис. 1.3):

- фазифікація вхідних величин, що входять до умов правил нечітких продукцій;

- агрегування підумов для правил, що мають складені антецеденти;

- активізація підвисновків;

- акумулювання висновків (для тих вихідних змінних, різні терми яких входять до різних активних правил нечітких продукцій, а зазвичай для кожної вихідної змінної так і є);

- дефазифікація вихідних змінних.

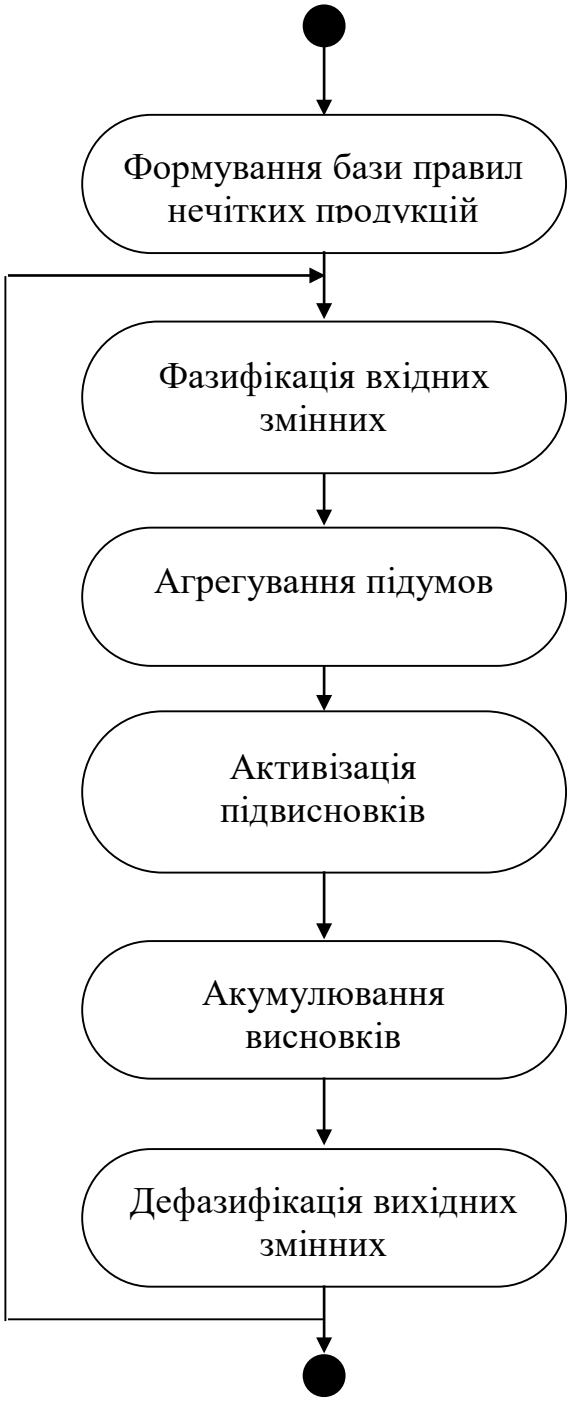


Рис. 1.3 Діаграма діяльності процесу нечіткого виводу
(у формі діаграми діяльності мови UML)

У цілому, можна констатувати, що нечітка логіка є ефективним інструментом для розв’язання слабко формалізованих задач, однак саме у галузі прогнозування технічно більш простим, але не менш ефективним з точки зору

достовірності результатів, що отримуються, є використання апарату нейронних мереж. Саме через відсутність необхідності розробки бази правил нечітких продукцій у даній роботі за основу буде взято нейронну мережу прямого поширення.

1.2.4 Аналіз можливостей застосування інших методів прогнозування для задачі, що розглядається

Очевидно, що існують й інші методи прогнозування, зокрема і такі, що можуть бути засновані на методах машинного навчання. Розглянемо ці методи в комплексі докладніше. Загалом, методи машинного навчання можна поділити на 4 класи (рис. 1.4).

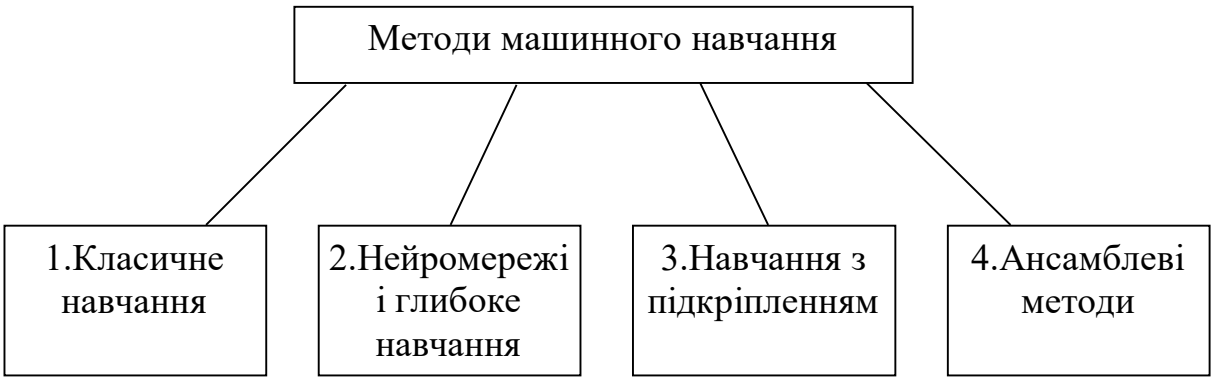


Рис. 1.4 Укрупнена класифікація методів машинного навчання

Перший із наведених класів можна розбити на навчання з учителем (побудова різноманітних класифікаторів: *k*-найближчих сусідів, Байєсівського, опорних векторів, дерев рішень, логістичної регресії; зведення регресій: лінійної, поліноміальної, гребеневої тощо) та без учителя (методи зменшення розмірності, пошук правил, кластеризація).

Другий клас, що на сьогодні представлений величезною кількістю наукових робіт у різних сферах, включає роботу із різними типами нейронних мереж.

До третього класу можна віднести генетичні алгоритми, Q-навчання, зведення глибоких Q-мереж.

Нарешті, до ансамблевих методів відносяться бустінг, стекінг та бегінг.

Ефективність усіх цих підходів буде різнитися, залежно від особливостей прикладної задачі, до розв'язання якої ці методи будуть застосовані. Однак, у будь-якому випадку математичний апарат нейронної мережі прямого поширення сигналів є досить простим (базується на лінійній згортці та вирахуванні лише одного значення нелінійної функції для кожного нейрона), а ефективність доведена результатами численних досліджень при прогнозуванні в інших предметних галузях. Відповідно, з метою раціонального використання ресурсів та з урахуванням наявності зручного ефективного інструменту, вирішено використовувати персептрон, як основу для розв'язання подальшої задачі прогнозування.

1.3 Формалізація процесу прогнозування результатів спортивних подій

Прогнозування в загальному сенсі полягає у визначенні майбутнього стану системи чи процесу $\vec{x}_{n+1}$ на основі наявних відомостей про попередні стани $\vec{x}_n$, $\vec{x}_{n-1}$, $\vec{x}_{n-2}$, …, $\vec{x}_k$ (причому сюди включаються як вхідні, так і вихідні змінні для кожного стану) та вхідної інформації про наступний стан, який прогнозується (наприклад, вектор його вхідних параметрів $\vec{u}_{n+1}$):

$$\vec{x}_{n+1} = f\left(\vec{x}_n, \vec{x}_{n-1}, \vec{x}_{n-2}, ..., \vec{x}_k, \vec{u}_{n+1}\right) \qquad (1.5)$$

де $\vec{x}_i$ – набір (вектор) параметрів, якими описується *i*-ий стан системи (процесу);

$\vec{x}_{n+1}$ – стан, що прогнозується, або, коротко, прогноз. У широкому сенсі він включає набір вхідних величин $\vec{u}_{n+1}$, що описують даний стан, та вихідних $\vec{v}_{n+1}$, які власне і є прогнозом, якщо говорити більш точно. Можна сказати, що:

$$\vec{x}_{n+1} = \vec{u}_{n+1} \cup \vec{v}_{n+1}; \qquad (1.6)$$

$\vec{x}_n$, $\vec{x}_{n-1}$, $\vec{x}_{n-2}$, …, $\vec{x}_k$ – попередні стани, повна інформація про які є відомою (а

саме, для кожного відомий набір вхідних та вихідних змінних окремо). Кількість таких станів позначимо $p = n - k + 1$;

$\vec{x}_k$ – найбільш віддалений у минулому часі стан, який ще враховується для побудови прогнозу;

$\vec{u}_{n+1}$ - набір значень для вхідних змінних системи (процесу), на основі яких обчислюватиметься набір вихідних даних $\vec{v}_{n+1}$, тобто формуватиметься набір $\vec{x}_{n+1}$ в цілому.

Характер інформації, яка являє собою прогноз $\vec{x}_{n+1}$, відповідає суті інформації $\vec{x}_n$, $\vec{x}_{n-1}$, $\vec{x}_{n-2}$, …, що є відомою про попередні стани.

Наприклад, якщо описується процес виготовлення деталі складної геометрії із певною кількістю отворів x_1 та паяних з'єднань x_2 (причому, зазвичай, x_1 і x_2 є досить великими числами), то час виготовлення деталі для певного конкретного обладнання складає x_3, а точність виготовлення (наприклад, по показнику маси) оцінюється відносним відхиленням x_4 (наприклад, у відсотках). В цілому *i*-тий стан системи можна описати четвіркою чисел:

$$\vec{x}_i = \{x_{i1}, x_{i2}, x_{i3}, x_{i4}\},$$

де весь вектор можна розбити на дві складові: $\vec{u}_i = \{x_{i1}, x_{i2}\}$ та $\vec{v}_i = \{x_{i3}, x_{i4}\}$, з яких перша уособлює вхідні параметри, а друга – вихідні. Власне прогнозом, є набір $\vec{v}_{n+1} = \{x_{(n+1)3}, x_{(n+1)4}\}$, але так само (у більш широкому сенсі) можна назвати і весь набір параметрів $\vec{x}_{n+1} = \{x_{(n+1)1}, x_{(n+1)2}, x_{(n+1)3}, x_{(n+1)4}\}$, тобто в тому числі і вхідних, якими описується прогнозований стан системи або процесу.

В загальному випадку кількість координат вектора, що описує *i*-тий стан може бути довільною, тому введемо для неї власне позначення *m*:

$$\vec{x}_i = \{x_{i1}, x_{i2}, ..., x_{im}\} = \{x_{ij} \mid j = 1..m\}. \qquad (1.7)$$

Розвиваючи (1.6) на усі стани $\vec{x}_i$, що беруть участь у розрахунках, формулу (1.7) можна подати ще і наступним чином:

$$\vec{x}_i = \vec{u}_i \cup \vec{v}_i = \{u_{ij} \mid j = 1..m_u\} \cup \{v_{ij} \mid j = 1..m_v\}, \qquad (1.8)$$

де m_u – кількість вхідних змінних, що описують стан системи;

m_v – кількість вихідних змінних, що описують стан системи (власне, це кількість величин, що мають бути передбачені, якщо говорити про скалярні величини, а не узагальнено про векторну величину $\vec{v}_i$).

Відмітимо, що

$$m = m_u + m_v. \qquad (1.9)$$

Елементи множин (векторів) $\vec{u}_i$ та $\vec{v}_i$ співпадають з деякими складовими вектора $\vec{x}_i$ (оскільки, власне, він утворюється об'єднанням цих двох векторів), що можна відобразити у наступних співвідношеннях:

$$u_{ij} \equiv x_{ij}\,|_{j=1..m_u}\ ;\ v_{ij} \equiv x_{i(m_u+j)}\,|_{j=1..m_v}. \qquad (1.10)$$

Якщо стан системи описується у лінгвістичних термінах, наприклад, «імовірність перемоги тенісиста *A* висока, якщо його опонент не дуже високого зросту», то і прогноз буде виконуватися у таких термінах.

Таким чином, структура та суть прогнозної інформації повністю відповідає параметрам інформації про попередні (відомі) стани системи.

Також потрібно пам'ятати, що для успішного розв'язання задачі прогнозування потрібним є набір даних не лише про попередні стани системи чи процесу, а й набір вхідних параметрів, що супроводжуватимуть той стан, що прогнозується. Аби підкреслити цю тезу, можна переписати (1.5) в дещо іншій формі з більш докладним урахуванням (1.6):

$$\vec{x}_{n+1} = f(\mathbf{X}, \vec{u}_{n+1}), \qquad (1.11)$$

де **X** – матриця, що містить інформацію про усі попередні стани системи, що ураховуються при прогнозуванні. Її можна записати наступним чином:

$$\begin{aligned} \mathbf{X} &= \{\vec{x}_n, \vec{x}_{n-1}, ..., \vec{x}_k\} = \{\vec{x}_i \mid i = n..k\} = \{\{x_{ij} \mid j = 1..m\} \mid i = n..k\} = \\ &= \{\{u_{ij} \mid j = 1..m_u\} \cup \{v_{ij} \mid j = 1..m_v\} \mid i = n..k\} \end{aligned} \qquad (1.12)$$

Матрицю **X** можна також описати у текстовій формі наступним чином: вона складається із рядків, кожен з яких є набором усіх координат вектора $\vec{x}_i$, тобто описує *i*-ий стан системи, який, відповідно, ураховується в прогнозній моделі. Перші m_u елементів кожного рядка являють собою значення вхідних параметрів для стану, який описується даним рядком. Наступні m_v елементів (тобто до самого кінця рядка, враховуючи (1.9)) є значеннями вихідних параметрів, що описували стан, який відповідає даному рядку.

Аналіз (1.11) дозволяє ввести просту, але ефективну класифікацію методів прогнозування за наявністю того, чи іншого аргументу в рівнянні (1.11). Всього можна виділити три випадки схем прогнозування:

1) коли враховується тільки інформація **X** про попередні стани, але не враховується наявна інформація $\vec{u}_{n+1}$ про вхідні параметри стану, який буде прогнозуватися:

$$\vec{x}_{n+1} = f(\mathbf{X}). \qquad (1.13)$$

Такий підхід означає вибудовування певного тренду, закону, по якому змінюються у часі (мається на увазі дискретний час, коли кожен новий момент просто відповідає утворенню нового стану системи) параметри стану. Підхід добре працює за відсутності суттєвих несподіваних збурюючих факторів, тобто при відносній стабільності зовнішнього середовища.

Наприклад, у країні з економікою, що активно розвивається, курс національної валюти зростає на кілька десятих долей відсотка щомісяця порівняно з курсом долара США. На основі такої інформації вже можна говорити про прогнозне значення курсу валюти через місяць, два, три, але все це дасть більш-менш точний прогноз лише за умови відсутності несподіваних глобальних потрясінь, таких як сильний землетрус, збройна агресія сусідньої держави, банкротство національних компаній чи найбільших корпорацій тощо.

Більш витончений приклад можна навести якраз з галузі прогнозування результатів спортивних подій. Шляхом ретельних спостережень можна

встановити, що команда «А» у своїй Національній лізі (де більшість команд мають приблизно однаковий рівень) виграє 2 матчі поспіль, а потім майже завжди програє один, після чого все повторюється. Така ситуація цілком природно може бути обумовлена тим, що після програшу більша частина команди перебуває у пригніченому стані і повертається до тренувань лише через кілька днів, накопичуючи сили. Під час же звичайних тренувань, навантаження, що дає тренер, є занадто важким для даної команди і гравці поступово виснажуються фізично, не маючи достатньо часу на відновлення. Запасу сил більшості гравців вистачає на термін між двома іграми, а на третю сил вже не залишається. Знаючи таку інформацію, можна досить точно зробити прогноз, за умови, що сила суперників у лізі є приблизно однаковою. При цьому результат буде отримано лише на основі **X**, без урахування інших, очевидно відомих, обставин $\vec{u}_{n+1}$ прогнозного поєдинку.

Для виявлення трендів використовують методи математичної статистики типу регресійного аналізу, сплайн-апроксимацію, перетворення Фур'є із подальшою фільтрацією тощо.

2) у наступній схемі організації прогнозної моделі враховуються тільки відомі обставини $\vec{u}_{n+1}$ майбутнього стану, але без залучення даних про історію попередніх станів **X**:

$$\vec{x}_{n+1} = f\left(\vec{u}_{n+1}\right). \qquad (1.14)$$

У галузі прогнозування результатів спортивних подій така ситуація є досить поширеною, а такі моделі називають *кваліметричними*. При цьому враховуються поточні значення різноманітних суттєвих показників, важливих для предметної галузі, що розглядається. Ці значення усереднюються, зважуються, згортаються і в результаті, звичайно, отримують якийсь один показник, за величиною якого можна зробити висновок про результат прогнозування.

З математичної точки зору для цього випадку застосовуються різноманітні методи згортки, нормалізації, методи експертного аналізу тощо.

3) нарешті, об’єднати переваги обох підходів та позбутися їх недоліків можна шляхом реалізації (1.11), де враховуються обидва типи інформації, наявної в системі.

1.4 Постановка задачі дослідження

Таким чином, у роботі потрібно провести розробку методу прогнозування результатів спортивних змагань з тенісу, причому у якості основи (математичного апарату) для розв’язання поставленої задачі потрібно використовувати нейронні мережі прямого поширення (персептрон).

Для реалізації методу розробити чіткий алгоритм із виділенням вхідної та вихідної інформації (її суті та структури), та здійснити його унаочнення.

Провести вибір (та надати його обґрунтування) щодо програмних засобів розробки, за допомогою яких потрібно звести програмну реалізацію розробленого методу. Засоби повинні відповідати загальноприйнятим концепціям у галузі технологій програмування.

Вимоги до програмної реалізації повинні бути стандартними, не потребувати використання спеціалізованої техніки (суперкомп’ютерів, потужних апаратних рішень, обчислювальних мереж тощо) та відповідати системним вимогам для сучасних операційних систем сімейства Windows.

Програма повинна мати графічний інтерфейс користувача та бути зручною у користуванні.

Потрібно провести тестування розробленого програмного продукту та оцінити його ефективність. Для цього потрібно зібрати масив статистичної інформації, частину якого використати для навчання мережі, частину для верифікації та підсумкового контролю.

Для створеного програмного продукту потрібно зробити мінімальний пакет супроводжувальної документації.

1.5 Висновки до розділу

У даному розділі проаналізовано проблему прогнозування у загальній постановці, а також – в контексті прогнозування саме результатів спортивних змагань. Проведено аналіз науково-технічних джерел, присвячених розв’язанню задачі прогнозування та встановлено, що найбільшу увагу наукова спільнота приділяє методам математичної статистики та машинного навчання, особливо на основі теорії нейронних мереж. Саме останній варіант взятий за основу для подальшого розв’язання задачі даної роботи. Описано вимоги до складу робіт та очікувані результати виконання дослідження (здійснено постановку задачі).

РОЗДІЛ 2. ПРОЕКТУВАННЯ АЛГОРИТМІВ ЗАСТОСУВАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ У ПРОГНОЗУВАННІ РЕЗУЛЬТАТІВ СПОРТИВНИХ ЗМАГАНЬ

2.1 Аналіз вхідних та вихідних даних проектованих алгоритмів

Усю інформацію, що буде використовуватися системою для прогнозування результатів тенісних матчів, можна розбити на дві великі категорії:

- основна інформація, яка є відомою до початку матчу;

- додаткова інформація, що виникає уже під час проведення матчу і може майже у режимі реального часу вводитися у систему прогнозування (процес відбувається вручну, тому, якщо говорити точно, використання терміну «у реальному часі» є не зовсім вірним).

У якості основної доцільно взяти інформацію про поточний стан першого та другого гравців, а саме:

- поточний номер у рейтингу ATP (для чоловіків) та WTA (для жінок) першого гравця;

- поточний номер у тому ж рейтингу другого гравця;

- наявність підтвердженої (наприклад, повідомленнями у новинах або спеціалізованих ЗМІ) травми у першого гравця за останні 30 днів (більшість мікротравм лікуються за менші періоди, а у побуті, як відомо, за цей період навіть лікуються переломи кінцівок, хоча у спорті після цього слідує тривалий період реабілітації, тому в даному випадку мова іде про травми, які не відміняють саму участь спортсмена у змаганні);

- наявність підтвердженої травми у другого гравця;

- тип покриття, що є дуже важливим показником, зважаючи на уподобання окремих гравців, і який, звичайно, будемо задавати числами («хард» = 1 як найбільш швидке покриття, трава = 0,66 як швидке покриття, грунт = 0,33 як повільне покриття, та синтетичне = 0 як найбільш повільне покриття);

- результати останніх матчів першого гравця та аналогічний показник для другого.

Останній показник розглянемо докладніше. По-перше, потрібно зафіксувати кількість останніх матчів, що беруться до уваги. Дуже мале значення, близьке до 1 (тобто 1-2), безперечно буде корисним (порівняно з повним ігноруванням інформації про результати попередніх матчів), але при цьому значно збільшується роль «випадку» і перемога чи поразка в одному минулому матчі може не відображати загальної картини готовності даного спортсмена. Навпаки, брати до уваги велику кількість попередніх матчів (7-10 і більше) не має сенсу, оскільки із кожним наступним матчем інформація про минулі змагання втрачає актуальність і не відображає реального рівня гри спортсмена. Оптимальною кількістю матчів будемо вважати $N \in [3, 5]$ і розробимо формулу, в якій кількість матчів, що враховуються, може бути змінною. Для цього максимальний результат, що відображає перемогу, приймемо на рівні 1, а поразку – на рівні 0. Тоді загальний результат за останні N матчів можна обчислити по формулі:

$$r = \frac{w_1 r_1 + w_2 r_2 + ... + w_N r_N}{w_1 + w_2 + ... + w_N}, \qquad (2.1)$$

де w_i – вага i-го результату, $i = 1..N$;

r_1 – найближчий у часі результат, який дорівнює 1, якщо гравець виграв, та 0 – якщо програв, r_2 – другий по віддаленості в часі результат, і т.д., r_N – найбільш віддалений (давній) результат.

Вагові коефіцієнти мають включати інформацію двох типів:

а) ступінь віддаленості в часі даного результату від поточного моменту, який пропонується впровадити за допомогою множника виду $w' = \frac{(N+1) - i}{N}$ у структурі цього вагового коефіцієнта. Таким чином, для найближчого результату

множник буде рівний 1, а для кожного наступного буде меншим на однакову відстань $\frac{1}{N}$ і для найбільш давньої події, що враховується, буде рівною $\frac{1}{N}$;

б) рівень суперника, з яким відбувався матч, що пропонується враховувати на основі його позиції *P* у світовому рейтингу (ATP чи WTA). При цьому суттєві відмінності у класі існують лише у перших позицій рейтингу, а ті спортсмени, що розміщуються на двозначних номерах, часто мають майже однакові показники майстерності. Так, різниця між першою і десятою ракеткою світу є незрівнянно більшою, аніж між 71 та 80-ою. Відповідно, у структурі вагового коефіцієнту впровадимо множник виду:

$$w'' = \begin{cases} \frac{1}{P} \\ 0{,}05 \end{cases}$$

Кінцево ваговий коефіцієнт будемо розраховувати по формулі:

$$w_i = w' \cdot w'' = \frac{(N+1)-i}{N} \cdot \max\left(\frac{1}{P_i}; 0{,}05\right) \qquad (2.2)$$

Відповідно, (2.1) можна подати у кінцевому вигляді, який і береться за основу у подальшій програмній реалізації для урахування результатів попередніх матчів даного гравця:

$$r = \frac{\sum_{i=1}^{N} \frac{(N+1)-i}{N} \cdot \max\left(\frac{1}{P_i}; 0{,}05\right) \cdot r_i}{\sum_{i=1}^{N} \frac{(N+1)-i}{N} \cdot \max\left(\frac{1}{P_i}; 0{,}05\right)}, \qquad (2.3)$$

Зважаючи на те, що усі $r_i \in \{0, 1\}$, розрахунки по формулі (2.3) надають число в інтервалі (0;1), отже, не потребують подальшої нормалізації на одиницю, яка є вкрай бажаною при використанні числового значення як входу нейронної мережі.

Окрім основної інформації, яка є відомою ще до початку змагання, відмінною рисою даного підходу є можливість урахування певної додаткової інформації про перебіг матчу, яка додається у систему прогнозування уже в процесі його проведення. У якості такої додаткової інформації, в першу чергу, можуть використовуватися результати першого сету. Як відомо, виграш одного з гравців може бути досягнутий уже наприкінці другого сету, тому відкладати введення інформації у систему на моменти пізніше перерви між 1 та 2 сетами немає сенсу.

Щодо структури цієї інформації можна зауважити наступне. Якщо загальну перемогу гравця у попередніх матчах з іншими суперниками доцільно оцінювати по системі 0/1, то в даному випадку корисніше впроваджувати в систему певну, більш детальну інформацію. Дійсно, перемога у першому сеті із рахунком 6:0 із ймовірністю близькою до 1 веде до перемоги першого гравця у всьому матчі, в той час, як при рахунку 6:4 перевага першого гравця є не дуже великою. Отже, краще всього задавати різницю у кількості виграних геймів двох гравців, що спостерігалася у першому сеті.

Всі вказані параметри (окрім сукупного результату попередніх матчів r) повинні бути нормалізовані на діапазон від 0 до 1, що є стандартним на практиці при подачі інформації на вхід нейронної мережі. Очевидно, що заборони на подачу сигналів з інших діапазонів не існує, однак в таких випадках процес навчання буде проводитися довше і може швидше приводити до перенавчання мережі.

Вихідну інформацію будемо шукати по системі 0/1, де 1 – виграш першого гравця, 0 – його програш (тобто виграш другого гравця).

Беручи до уваги наведені дані, можна переходити до процесу розробки алгоритмів, що на основі цих даних будуть проводити прогнозування переможця матчу.

2.2 Розробка алгоритму прогнозування результатів спортивних змагань на базі методів машинного навчання

Проектована система, як і будь-які програмні продукти, реалізує певні алгоритми, що можуть описувати різні рівні абстракції роботи програми. В першу чергу, програмне забезпечення реалізує алгоритм роботи користувача системи із даним програмним продуктом, який є досить простим (пакетним або лінійним) і наведений на рис. 2.1.

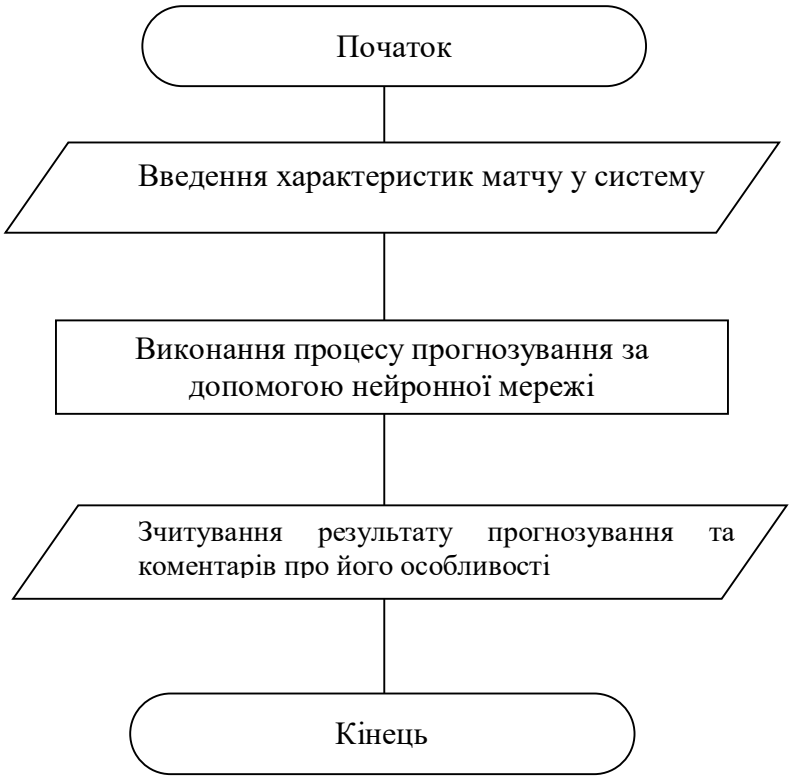


Рис. 2.1 Алгоритм роботи користувача з системою прогнозування результатів спортивних змагань

Очевидно, що для роботи із програмою відповідно до алгоритму (рис. 2.1), нейронна мережа уже повинна бути навченою, тобто ваги її синапсів повинні бути вже визначені і зберігатися у постійному запам’ятовуючому пристрої ПК (у файлі на диску) або принаймні в оперативній пам’яті завантаженої програми. Очевидно, що налаштування мережі має виконувати ця ж сама програма (хоча

принципово це можна робити і окремою програмою, але такий підхід не є логічним, адже краще реалізувати один повноцінний програмний продукт, присвячений певній прикладній задачі, аніж декілька простіших). Єдиним обмеженням для виконання процесу тренування можна зробити на один рівень більш глибоко розміщений елемент управління для активізації процесу навчання (наприклад, процес прогнозування робити безпосередньо у головному вікні програми, яке видно уже при її завантаженні, а процес навчання мережі розмістити в окремому вікні, що викликається за допомогою спеціального пункту меню).

Алгоритм процесу навчання можна зробити наступним. В окремому каталозі TrainingData розмістити декілька окремих файлів із даними для навчання. Кількість файлів може бути рівною числу матчів, що використовуються для навчання, і кожен файл присвячено одному матчу. У файлі розміщуються контрольовані параметри матчів (які розглянуто у попередньому підрозділі), а також реально отриманий результат цього матчу. При такій організації вхідної інформації алгоритм навчання мережі буде таким, як зображено на рис. 2.2.

На рис. 2.2 схема процесу показана укрупнено, хоча більшість її елементів мають порівняно просту деталізацію. Найскладнішою є реалізація елементу «Відкорегувати ваги синапсів мережі за методом зворотного поширення помилки», який відповідає однойменному загальновідомому алгоритму.

Також на схемі (рис. 2.2) не показано прості елементи, на зразок вибору максимального значення із набору похибок. Самі ж похибки у випадку векторного виходу (багатовимірного, із кількістю вихідних нейронів, більшою 1) нейронної мережі можуть обчислюватися за різними процедурами, тому в загальному випадку ця дія також може бути нетривіальною, однак у нашому випадку нейронна мережа має всього лише один вихід, що значно спрощує

обчислення похибки, яка у такому випадку береться як модуль арифметичної різниці прогнозованого результату матчу та реально отриманого значення.

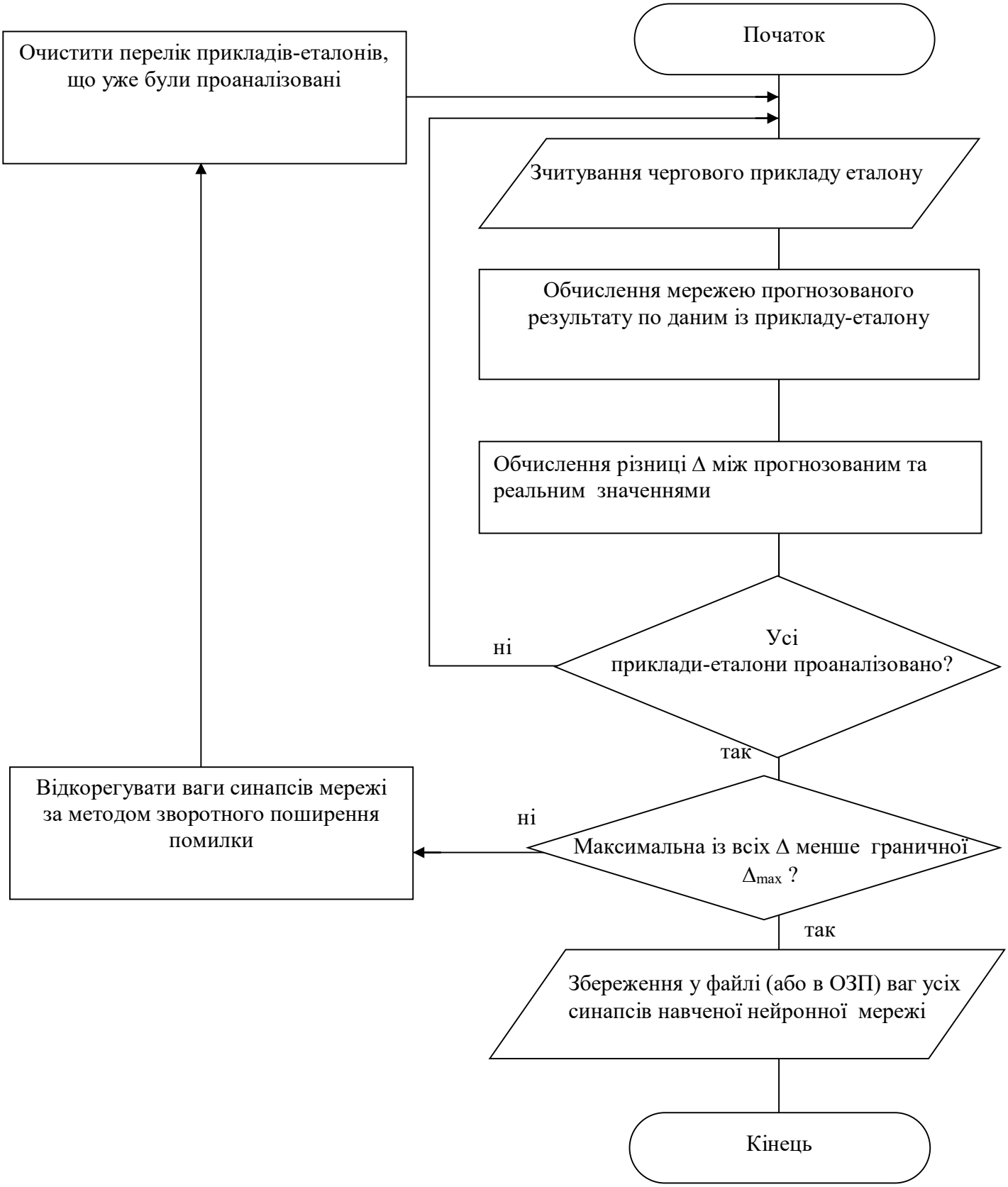


Рис. 2.2 Блок-схема алгоритму процесу навчання нейронної мережі

Розглянувши детально алгоритми, що будуть покладені в основу роботи системи, можна переходити до моделювання її основних характеристик.

2.3 Моделювання системи прогнозування результатів спортивних змагань на базі методів машинного навчання

2.3.1 Структурна модель

У першу чергу, потрібно визначити складові проектованої системи та зв’язки, що мають існувати між ними (рис. 2.3).

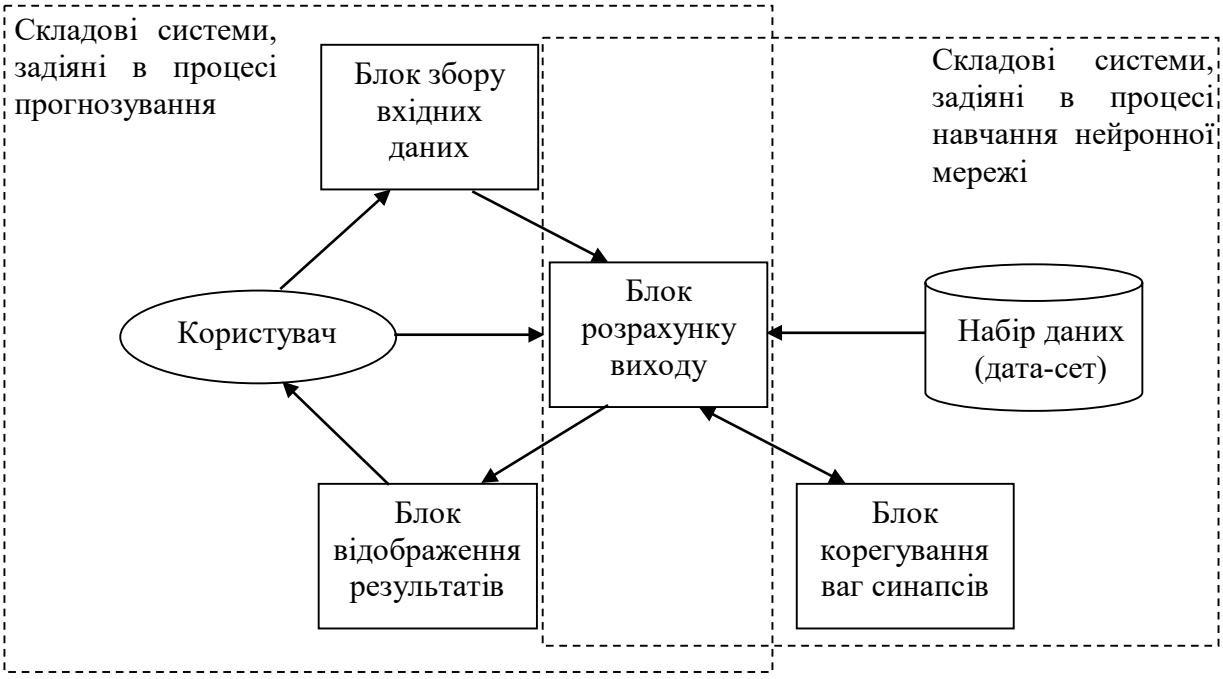


Рис. 2.3. Структурна модель системи прогнозування результатів спортивних змагань на базі нейронної мережі.

У системі виділяємо:

- блок збору та підготовки вхідних даних (у ньому користувач вводить параметри майбутнього прогнозованого матчу);

- блок розрахунку вихідного значення (власне, прогнозу), який реалізує пряме проходження сигналу через нейронну мережу;

- блок відображення результатів користувачеві, який повинен надати результат прогнозування у зрозумілій формі для людини, яка не є фахівцем з машинного навчання.

Ці три блоки відносяться до звичайного робочого процесу використання системи (тобто процесу прогнозування). Однак, перед тим потрібно провести навчання мережі, для чого використовуються:

- набір даних для тренування системи (дата-сет);

- блок розрахунку вихідного значення (очевидно, потрібно використати той самий, що уже був згаданий у попередньому переліку);

- блок корегування ваг синапсів мережі, який реалізує один із алгоритмів навчання з учителем, наприклад, найбільш популярний – метод зворотного поширення помилки (back-propagation).

Таким чином, структуру системи можна вважати встановленою.

2.3.2 Інформаційна модель

Тут розглянемо, які джерела інформації будуть наявними у системі, що розробляється (рис. 2.4).

Рис. 2.4. Інформаційна модель системи прогнозування

У першу чергу, невід'ємним елементом роботи системи прогнозування є набір усіх вхідних даних, що будуть використовуватися під час процесу тренування нейронної мережі (іншими словами – дата-сет). На основі цієї інформації за допомогою складових системи, задіяних у процесі навчання нейронної мережі, генеруються таблиці (набір) вагових коефіцієнтів (ваг) усіх зв'язків (синапсів) між нейронами мережі. Далі, на основі цих таблиць, а також на основі набору вхідних даних, що описують прогнозований матч (і які отримуються від користувача системи), за допомогою складових системи, задіяних у процесі прогнозування, вираховується кінцеве прогнозоване значення (результат матчу). Інша інформація системи має другорядне значення і у даному контексті може не розглядатися.

2.3.3 Функціональна модель

Нарешті, необхідно описати функціональність системи, тобто створити її функціональну модель. Очевидно, що існують різні способи для опису та унаочнення функціональних моделей, однак найбільш простим із них є текстовий опис. Отже, проектована система повинна забезпечувати реалізацію наступних дій (рис. 2.5):

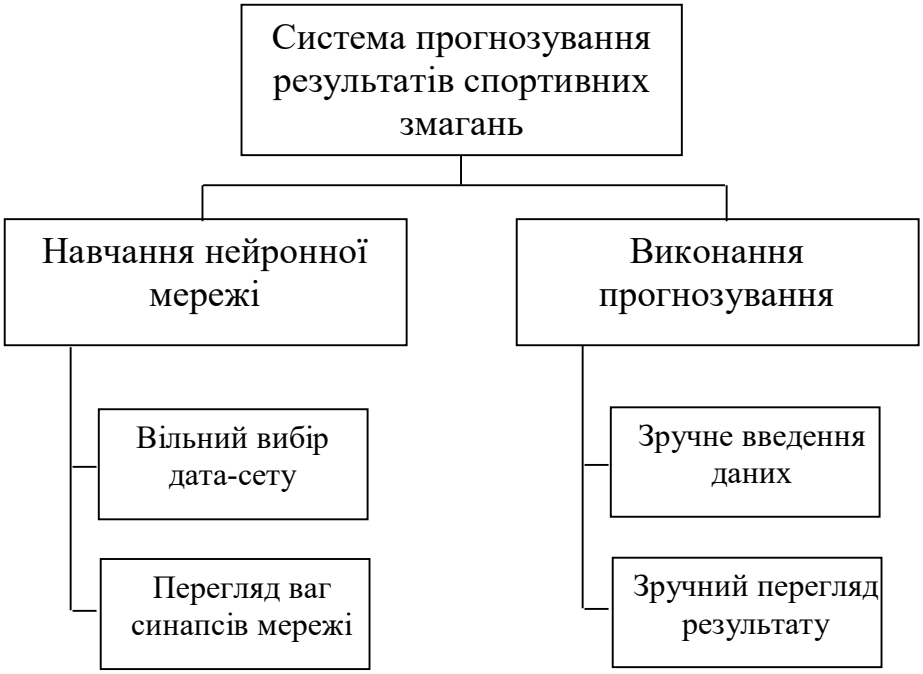


Рис. 2.5. Перелік функцій проектованої системи прогнозування

- дозволяти задання різних файлів для тренування системи (свобода у виборі дата-сету користувачем);

- дозволяти перегляд вагових коефіцієнтів для будь-яких синапсів нейронної мережі;

- проводити процес введення вхідних даних для прогнозування у зручному, візуальному режимі;

- отримання результату користувачем у зручному, візуальному режимі.

2.4 Висновки до розділу

Таким чином, у даному розділі розроблено алгоритмічне підґрунтя для побудови подальшої практичної реалізації системи прогнозування результатів спортивних змагань на основі нейронної мережі. По-перше, встановлено тип і структуру вхідних даних, що повинні поступати на вхід мережі. На відміну від існуючих рішень, окрім інформації, яка є відомою ще перед початком змагання, запропоновано використовувати також результати завершеної частини змагання (зокрема, першого сету тенісного матчу) для прогнозування загального результату гри.

У розділі розроблено алгоритми роботи проектованої системи, причому як для нормального режиму її використання (безпосередньо виконання прогнозування), так і для її налаштування – навчання нейронної мережі; наведено відповідні схеми алгоритмів. Розроблено моделі проектованої системи, що дозволяють перейти до процесу її програмної реалізації.

РОЗДІЛ 3. ЗДІЙСНЕННЯ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ ПРОГНОЗУВАННЯ РЕЗУЛЬТАТІВ СПОРТИВНИХ ЗМАГАНЬ НА БАЗІ МЕТОДІВ МАШИННОГО НАВЧАННЯ

3.1 Вибір засобів розробки

3.1.1 Вибір технологій програмування

Першочерговим питанням, яке постає перед розробниками будь-якого програмного забезпечення, є вибір моделі або технології, парадигми його розробки [22]. Такими, що широко використовуються на сьогоднішній день у виробничій практиці, є технології структурного (процедурного) та об'єктно-орієнтованого програмування. Кожна з них має свої особливості, переваги і недоліки, які розглянемо докладніше.

Структурне [23], або як його ще називають, процедурне програмування базується на використанні окремих структурних блоків, в першу чергу, підпрограм (процедур і функцій) і передбачає побудову програми відповідно до трьох основних принципів: слідування, розгалуження, повторення.

На зміну структурному поступово прийшло об'єктно-орієнтоване програмування (деякі сучасні мови програмування навіть не дозволяють створити структурну програму, тільки об'єктно-орієнтовану – як, наприклад, C# чи Java). Проте, при створенні невеликих програм (наприклад, до 10000 рядків коду і без передбачуваного розширення) застосування цієї методики програмування більш виправдано і код краще сприймається, ніж його об'єктно-орієнтований варіант.

Суть же методології об'єктно-орієнтованого програмування [24] полягає в тому, що система розглядається, як сукупність окремих сутностей- об'єктів, які мають набір якихось своїх внутрішніх параметрів-властивостей, а також можуть взаємодіяти між собою. Реалізовуючи всі сутності, необхідні, згідно з алгоритмом, для роботи програми, у вигляді класів і об'єктів, спрощується її

розуміння. Саме тому ОО-підхід рекомендується до застосування для великих проектів. Можна сказати, що розбиття програми на об’єкти і проектування їх класів наближає розуміння предметної області до звичного людського образу мислення.

Крім розбиття (декомпозиції) всієї предметної області на об’єкти (класи) і співвідношення між ними, також ОО-підхід передбачає дотримання трьох основних його принципів: інкапсуляція, наслідування та поліморфізм.

Враховуючи особливості наведених методів програмування, вибираємо об’єктно-орієнтований підхід як більш сучасний, гнучкий та прогресивний, що відповідає середнім масштабам проектованого ПЗ.

3.1.2 Вибір мови програмування

На сьогодні існує багато мов програмування, які підтримують обрану у попередньому підрозділі об’єктно-орієнтовану парадигму (у відносно повній мірі, тобто з поліморфізмом та іншими більш сучасними нововведеннями). Із популярних мов можна виділити C++, Java, C#.

Перший варіант, мова C++, є надзвичайно потужною [25], має стабільну популярність, але є досить складною для сприйняття; фактично, це інструмент для професіоналів найвищого рівня. Більш популярними серед широкої спільноти програмістів середнього рівня є мови Java та C#.

Таким чином, при реальному процесі вибору мови програмування для професійної розробки на сьогоднішній день потрібно розглядати дві укрупнені альтернативи: мова C++ з одного боку та одна з мов, Java чи C#, з іншого. Значну кількість полеміки було присвячено порівнянню цих двох мов, але беззаперечним фактом є значна перевага продукту від Microsoft над конкурентом у частині зведення користувацького інтерфейсу. У цьому питанні середовище Visual Studio надає дуже високий рівень зручності. Отже, зважаючи на необхідність створення

програмного продукту для операційної системи Windows, доцільніше використовувати C#, а не Java.

Якщо ж порівнювати C++ та C#, то перша мова є більш пристосованою для системних задач (як от організація зв'язку по TCP-IP, численні математичні операції, відслідковування подій засобів введення-виведення тощо). У результаті порівняльного аналізу обираємо мову програмування C++, що найкраще відповідає постановці задачі даного дослідження, наведеній у кінці попереднього розділу. Коротко розглянемо особливості цієї мови програмування.

Якщо говорити про характеристики мови програмування C++, на якій буде розроблятися проектована система, то потрібно сказати, що вона є високорівневою компільованою мовою загального призначення зі строгою типізацією, яка підходить для створення різноманітних додатків, починаючи від настільних програм, до веб-додатків і, навіть, мобільних програм. Беручи до уваги те, що на C++ можна впроваджувати «асемблерні» вставки, можна сказати, що цією мовою можливо реалізувати практично будь-який програмний продукт. Таким чином, суттєвим плюсом мови C++ є її універсальність в частині створення додатків для різноманітних цільових платформ.

Суттєвою перевагою мови C++ [26] є наявність стандартної бібліотеки шаблонів (Standard Templates Library, далі – STL), у якій зведені разом усі популярні структури даних (створені у вигляді окремих класів), а також реалізовані базові алгоритми роботи з цими структурами. Зокрема, як відомо, основними алгоритмами є пошук та сортування – ці, та багато інших операцій, присутні для усіх структур даних бібліотеки STL. Серед таких структур можна виділити:

- масив (array);
- вектор (vector);
- список (list);
- дека, або черга (deque);

- хеш (hash) тощо.

Беручи до уваги усі переваги мови C++, вона прийнята за основну для виконання програмної реалізації у даному дослідженні.

3.1.3. Вибір середовища розробки та допоміжного програмного забезпечення.

Перейдемо до розгляду конкретних програмних засобів, які можуть використовуватися для процесу розробки на обраній мові C++. Основним із них, що дозволяє реалізовувати ОО-проекти цією мовою, є середовище розробки Microsoft Visual Studio. Дане середовище – це серія продуктів компанії Майкрософт, які включають в себе інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight. Важливим фактом є те, що у цьому середовищі присутні безкоштовні версії, що дуже зручно для законної розробки у ньому некомерційних програмних продуктів, зокрема таких, що виконуються в рамках магістерських досліджень.

Ще одним середовищем розробки для мови C/C++ є Borland C++ Builder (з 2008 року – просто C++ Builder). Це середовище [27] є дуже зручним, але через слабку маркетингову політику компанії-розробника (Embarcadero Technologies) значна частка програмістів вважає, що даний продукт не використовується ще з початку XXI ст. Характерною особливістю цього середовища розробки (як і усіх програмних продуктів, що розроблялися компанією Borland) була і є надзвичайна простота у вирішенні деяких складних (але рутинних) питань, які постають перед будь-якими розробниками мовою C++. Головним плюсом цього середовища

розробки є автоматизація задачі створення ефективного інтерфейсу користувача програми. Також можливим є використання одного із компіляторів мови C/C++, що підтримуються open-source спільнотою програмістів, наприклад GCC, або DJ++ Compiler. Однак ці продукти (як і велика кількість інших продуктів, пов'язаних з цим об'єднанням) характеризуються складністю роботи з ними, що у комплексі спричиняє суттєві втрати робочого часу на виконання рутинних дій, які у тому ж C++ Builder виконуються за лічені хвилини.

Таким чином, зважаючи на усі переваги, у якості основного інструменту для подальшої розробки обираємо інтегроване середовище розробки C++ Builder. Вибір в основному обумовлений великою швидкістю розробки візуального інтерфейсу в даному середовищі. Створення макета вікна програми (форми, по термінології Borland) відбувається в наочному, інтуїтивно зрозумілому режимі.

Ще однією перевагою продуктів від Borland є наявність значної кількості компонентів (тобто класів) та їх бібліотек, за допомогою яких можна зручно розв'язувати високорівневі завдання досить ефективними методами.

Таким чином, використання середовища розробки C++ Builder дозволяє поєднувати з одного боку швидкість розробки в стилі компонентного програмування (особливо, в частині створення призначеного для користувача інтерфейсу), а з іншого – гнучкі, як завгодно низькорівневі, можливості мови C/C++.

Досить часто в процесі реалізації того, чи іншого програмного продукту, виникає потреба у використанні додаткового програмного забезпечення. Тому обов'язковим етапом процесу проектування ПЗ є дослідження питання необхідності залучення інших інструментів, окрім основного, яким, зазвичай, є інтегроване середовище розробки (C++ Builder, який було обрано у попередньому пункті).

У даній роботі дане питання є особливо актуальним, оскільки у якості базового інструменту для розв'язання поставленої задачі прогнозування обрано

нейронну мережу, яка є досить складною системою і має певну специфіку. Вище розглянуто загальні етапи роботи з нейронними мережами, зокрема, і такий, як її навчання. Даний процес може виконуватися різними шляхами, але усі вони без виключення є не простими з алгоритмічної точки зору. Так само не дуже простим (хоча і простішим за навчання) є безпосередньо саме використання мережі (тобто процес визначення вихідного сигналу на основі вхідних значень).

Відповідно, виникає питання, яке прямим чином відноситься до процесу проектування: можливо потрібно ці не прості алгоритми запозичити із сторонніх бібліотек? Для відповіді на нього потрібно розглянути наявні розв'язки (чи, принаймні, найбільш характерні з них), що дозволяють розв'язувати задачі, пов'язані із нейронними мережами. Таким програмним продуктом є зручний і потужний фреймворк AForge.net, добре відомий серед професіоналів, які розв'язують широке коло алгоритмічних задач. На офіційному сайті сказано, що цей продукт був створений для розробників програмного забезпечення у галузі машинного зору та штучного розуму, а саме:

– обробки зображень;

– нейронних мереж;

– генетичних алгоритмів;

– машинного навчання;

– робототехніки;

– тощо.

Цю бібліотеку потрібно встановлювати у систему окремо (перед тим – завантажити з офіційного сайту http://aforgenet.com та встановити в операційну систему Windows), а потім підключити до проекту за допомогою вікна його налаштувань. Як бачимо із наведеного переліку, даний програмний продукт є надзвичайно потужним, оскільки включає широкий перелік можливостей. Якраз в універсальності та значній наповненості і полягає недолік даного ПЗ: у програмному продукті, що має вузьку направленість і призначений строго для

прогнозування результатів спортивних змагань, включати можливості машинного зору, генетичних алгоритмів та робототехніки є надлишковим підходом, що перевантажить програму (підвищить системні вимоги, зробить її роботу менш зручною і повільнішою).

Крім того, при використанні у будь-якому програмному продукті довільних сторонніх бібліотек у розробників завжди виникає питання, чи адекватно вони виконують свою функцію (чи правильним є результат на усіх без винятку наборах вхідних даних, чи не є занадто великою похибка, з якою представляється результат, на деяких наборах вхідних даних тощо). Без детального аналізу вихідного коду відповісти на ці питання однозначно не можливо, але велика кількість дійсно функціонально привабливих бібліотек мають закритий вихідний код. В той же час, якщо задачі, що виконуються бібліотекою, є дійсно алгоритмічно складними, то навіть за умови відкритості вихідних текстів бібліотеки, трудомісткість перевірки буде у відносних одиницях наближатися до трудовитрат на розробку такого коду власними силами.

Висновок, що можна зробити із наведених міркувань, полягає у тому, що використовувати сторонні бібліотеки потрібно тільки за умови, що розробити відповідний код власними силами практично неможливо в існуючих при розробці ПЗ часових та фінансових обмеженнях. У даній роботі найскладнішим у реалізації є алгоритм навчання нейронної мережі, однак і його можна порівняно легко реалізувати силами одного розробника. Відповідно, питання доцільності використання сторонніх бібліотек, яке обов’язково потрібно було розглянути в процесі проектування програмного продукту, саме у даному випадку вирішується негативно: додаткового програмного забезпечення використовувати не потрібно, усі алгоритми доцільно реалізувати власноруч у відповідному монолітному програмному продукті.

3.2 Розробка інтерфейсу користувача системи

Одним із перших кроків при створенні нового програмного забезпечення є проектування його інтерфейсу користувача. Для програмних продуктів, які працюють у пакетному режимі (як даний програмний продукт, який спочатку збирає усі вхідні дані, потім виконує процес розрахунку, а кінцево відображає отриманий результат (рис. 3.1)), у головному вікні програми доцільно розмістити відразу усі елементи управління для зчитування вхідних даних, що потрібні для запуску процесу прогнозування. Тут же має розміщуватися елемент (очевидно – кнопка «Розрахувати») для запуску процесу розрахунків, які виконуються при прогнозуванні. Після завершення обчислень, результати прогнозування потрібно відобразити користувачеві, що можна виконувати у новому вікні, або у тому ж самому головному, якщо кількість елементів управління для введення інформації не є великою.

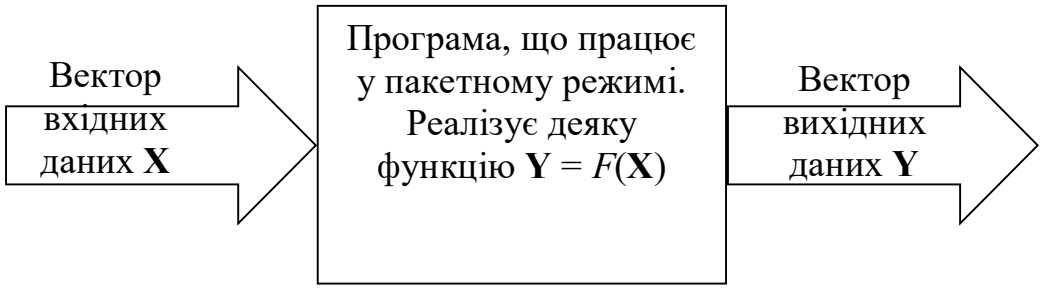


Рис. 3.1 Схема роботи пакетної програми

Таким чином, потрібно визначитися із точним набором вхідних величин, які впливають на результати тенісного матчу, до складу якого у даній роботі запропоновано наступні:

1) поточний номер у рейтингу ATP (для чоловіків) та WTA (для жінок) першого гравця;

2) поточний номер у тому ж рейтингу другого гравця;

3) наявність підтвердженої травми у першого гравця за останні 30 днів (0 або 1);

4) наявність підтвердженої травми у другого гравця (0 або 1);

5) тип покриття: хард = 1, трава = 0,66, грунт = 0,33, синтетичне = 0;

6) результат останнього матчу, зіграного першим гравцем (0 або 1);

7) результат передостаннього матчу, зіграного першим гравцем (0 або 1);

8) результат матчу перед передостаннім, зіграного першим гравцем (0 або 1);

9) результат останнього матчу, зіграного другим гравцем (0 або 1);

10) результат передостаннього матчу, зіграного другим гравцем (0 або 1);

11) результат матчу перед передостаннім, зіграного другим гравцем (0 або 1);

12) різниця у кількості виграних геймів після першого сету (від -6, що означає розгромний програш першого гравця, до +6, що означає розгромну перемогу першого гравця).

Таким чином, у головному вікні реалізуємо 12 елементів управління із підписами. Окремо розташовуємо кнопки «Навчити мережу» та «Розрахувати». Результатом прогнозування є одне число, тому можна розмістити відповідний елемент управління у тому ж вікні, де і згадані поля для введення вхідних даних (рис. 3.2).

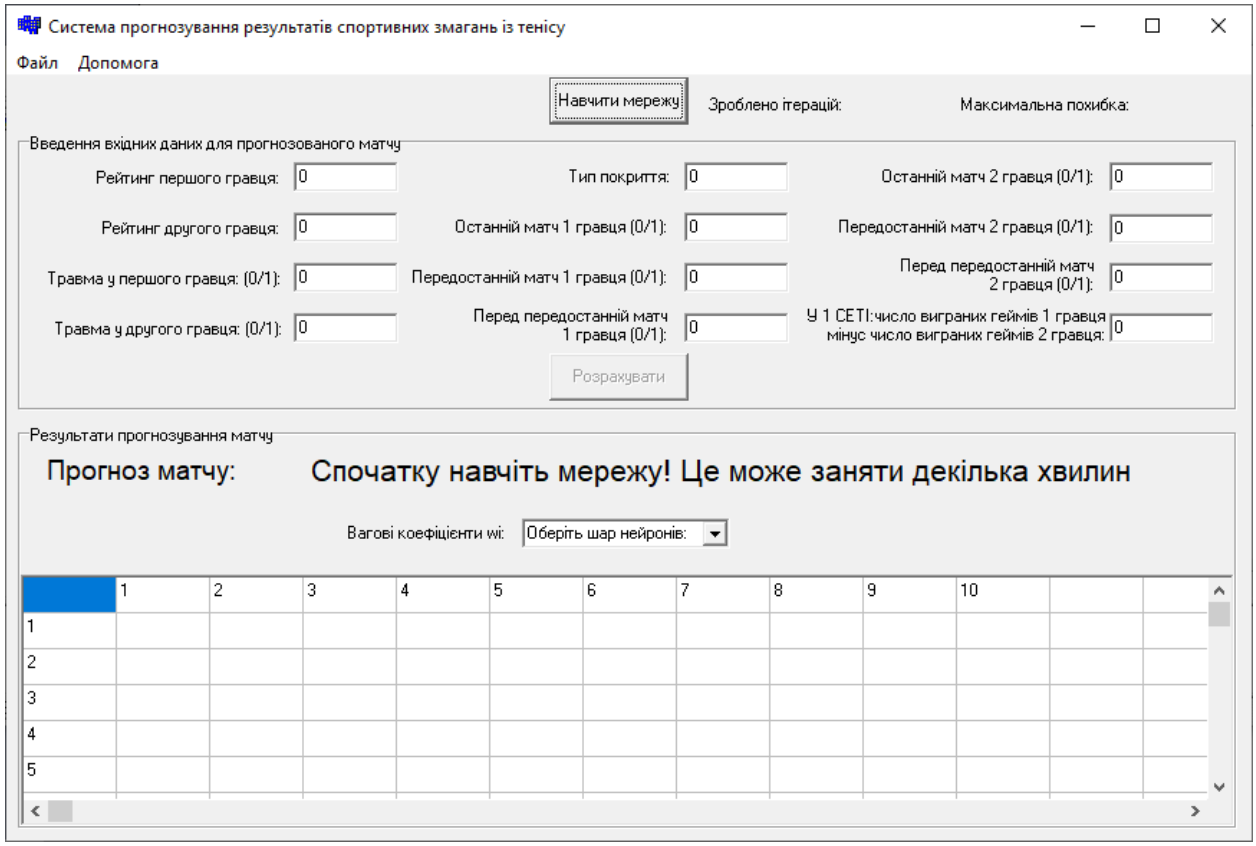


Рис. 3.2 Інтерфейс користувача системи прогнозування результатів змагань із тенісу

Передбачаємо деякі додаткові елементи управління, що відображують певну сервісну інформацію, що може бути корисною з точки зору досліджень системи, а саме:

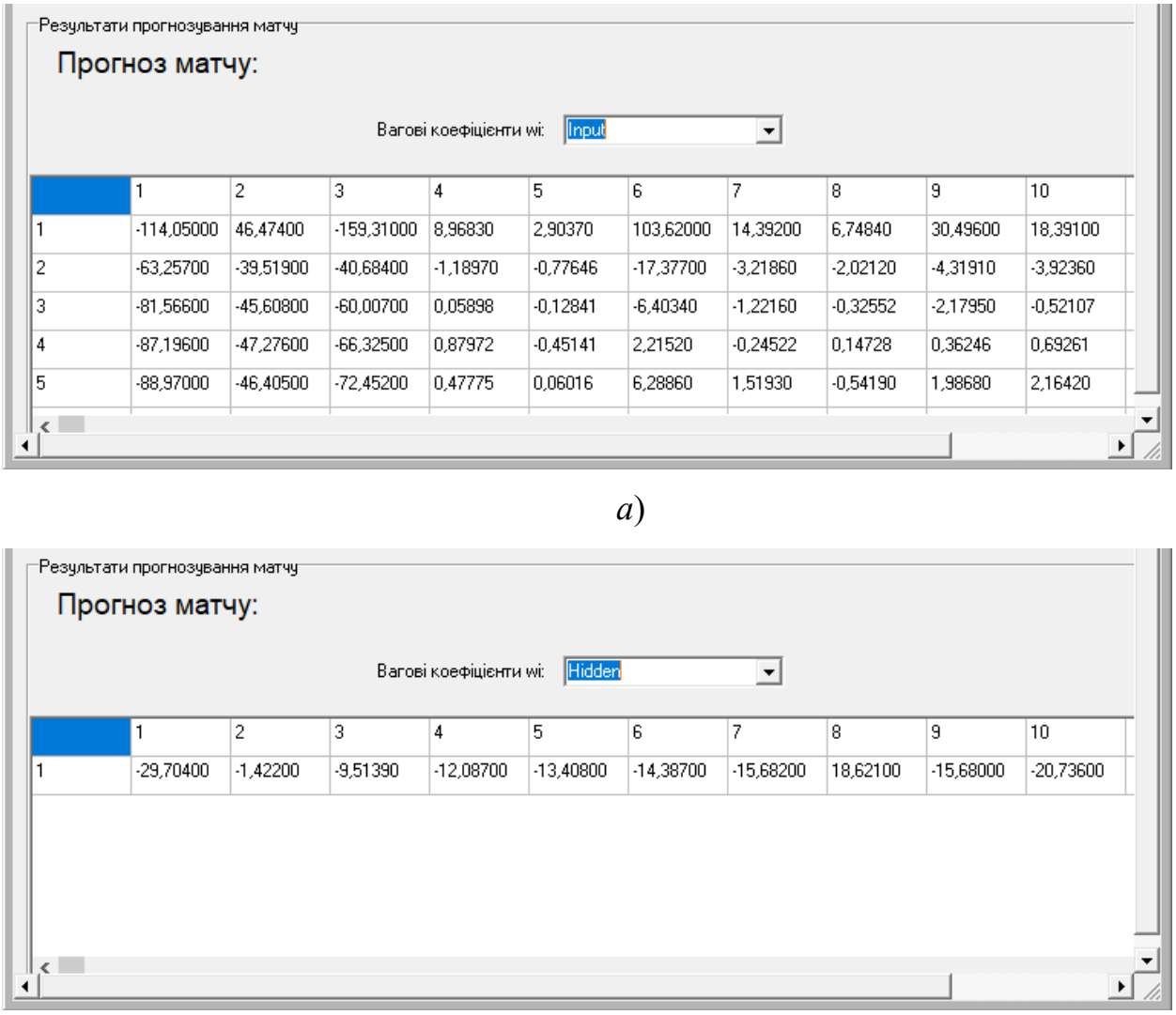
– кількість виконаних ітерацій (епох) при навчанні мережі за методом зворотного поширення помилки;

– максимальне значення похибки, отриманої під час навчання в кінці цього процесу (максимальне серед усіх елементів дата-сету);

– вагові коефіцієнти w_i (зв'язків між нейронами), що отримано в результаті навчання.

Останній пункт потрібно розглянути докладніше, адже кількість нейронів у різних шарах буде різною. Тому розмір елементу управління StringGrid (таблиця у якій виводяться ваги) має динамічно змінюватися залежно від

обраного пункту у випадаючому списку і початково встановлений у позицію «Оберіть шар нейронів». Коли користувач змінює значення цього поля, відповідно змінюється і розмір таблиці з коефіцієнтами (рис. 3.3).



	1	2	3	4	5	6	7	8	9	10
1	-114,05000	46,47400	-159,31000	8,96830	2,90370	103,62000	14,39200	6,74840	30,49600	18,39100
2	-63,25700	-39,51900	-40,68400	-1,18970	-0,77646	-17,37700	-3,21860	-2,02120	-4,31910	-3,92360
3	-81,56600	-45,60800	-60,00700	0,05898	-0,12841	-6,40340	-1,22160	-0,32552	-2,17950	-0,52107
4	-87,19600	-47,27600	-66,32500	0,87972	-0,45141	2,21520	-0,24522	0,14728	0,36246	0,69261
5	-88,97000	-46,40500	-72,45200	0,47775	0,06016	6,28860	1,51930	-0,54190	1,98680	2,16420

а)

	1	2	3	4	5	6	7	8	9	10
1	-29,70400	-1,42200	-9,51390	-12,08700	-13,40800	-14,38700	-15,68200	18,62100	-15,68000	-20,73600

б)

Рис. 3.3 Відображення вагових коефіцієнтів нейронної мережі: *а* – для синапсів між вхідним шаром та прихованим; *б* – між прихованим та вихідним

В момент одразу після завантаження програми кнопка «Розрахувати» є неактивною, оскільки мережу потрібно навчити на тренувальній вибірці. Після натискання на кнопку «Навчити мережу» деякий час відбувається процес навчання, після якого розрахування стає доступним, а навчання навпаки блокується.

Окрім головного вікна, реалізуємо ще одну форму, яка відображатиме традиційне вікно «Про програму…» (рис. 3.4).

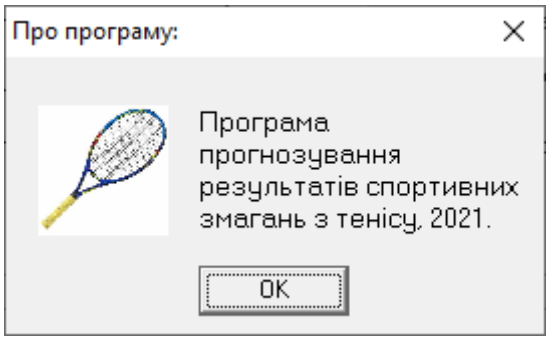


Рис. 3.4 Вікно «Про програму»

Визначивши деталі створення інтерфейсу користувача програми, можна переходити до опису деталей програмної реалізації окремих алгоритмічних складових проектованого програмного забезпечення.

3.3 Особливості програмної реалізації найбільш суттєвих елементів розробленого алгоритму прогнозування

Як уже було встановлено вище, за основу системи прогнозування взято багатошаровий персептрон. Для вказаних цілей цілком достатньо трьох шарів:

– вхідного шару (це умовний шар, для виходів вузлів якого не застосовуються функції активації і який необхідний лише для організації можливості подання вхідних сигналів на усі нейрони прихованого шару);

– одного прихованого шару;

– вихідного шару.

Дослідимо характеристики цих шарів.

На вхідний шар подаються одновимірні вектори, які формуються на основі набору значень дванадцяти вказаних вище показників матчу. На їх основі й виконуватиметься прогнозування. Кількість нейронів вхідного шару приймаємо

рівною кількості вхідних змінних, тобто 12. У програмній реалізації для її більшої універсальності число нейронів у вхідному шарі задаємо у вигляді константи N:

```
const N=12; //кількість нейронів вхідного шару
```

Прихований шар для складних задач може містити більшу кількість нейронів, ніж вхідний та вихідний шари. В нашому випадку кількість вхідних змінних рівна 12, а вихідних змінних взагалі одна (причому для її позначення у програмі будемо для універсальності використовувати константу `const M=1;`), тобто за числом входів та виходів мережа не може вважатися складною. Важливу роль у визначенні структури нейронної мережі, а саме, кількості нейронів прихованих шарів, грає обсяг навчальної вибірки: чим більшій кількості ситуацій потрібно навчити мережу, тим більше вона повинна містити нейронів. У нашому випадку кількість випадків для навчання мережі складає 214. Ці фактори говорять про те, що кількість нейронів прихованого шару можна не робити великою, а просто рівною кількості нейронів у вхідному шарі (тобто $N = 12$, також у якості експерименту була протестована робота нейронної мережі за N=8 та N=18).

Найпростішим у випадку, що розглядається, є вихідний шар: оскільки прогнозуванню підлягає лише одна величина (прогноз результату матчу, а саме його переможця), то і вихід мережі буде один. Отже, кінцеве число шарів у персептроні буде рівним 3 (причому один вхідний шар є, фактично, фіктивним, бо його виходи просто повторюють вхідні сигнали без застосування функції активації), а число нейронів у них буде 12-12-1. Кількості нейронів встановлюємо у масиві Nneuronsinlayer:

```
Nneuronsinlayer[0]=N;
Nneuronsinlayer[1]=N;
Nneuronsinlayer[2]=M;
```

Структура проектованої нейронної мережі тоді приймає вид, показаний на рис. 3.5.

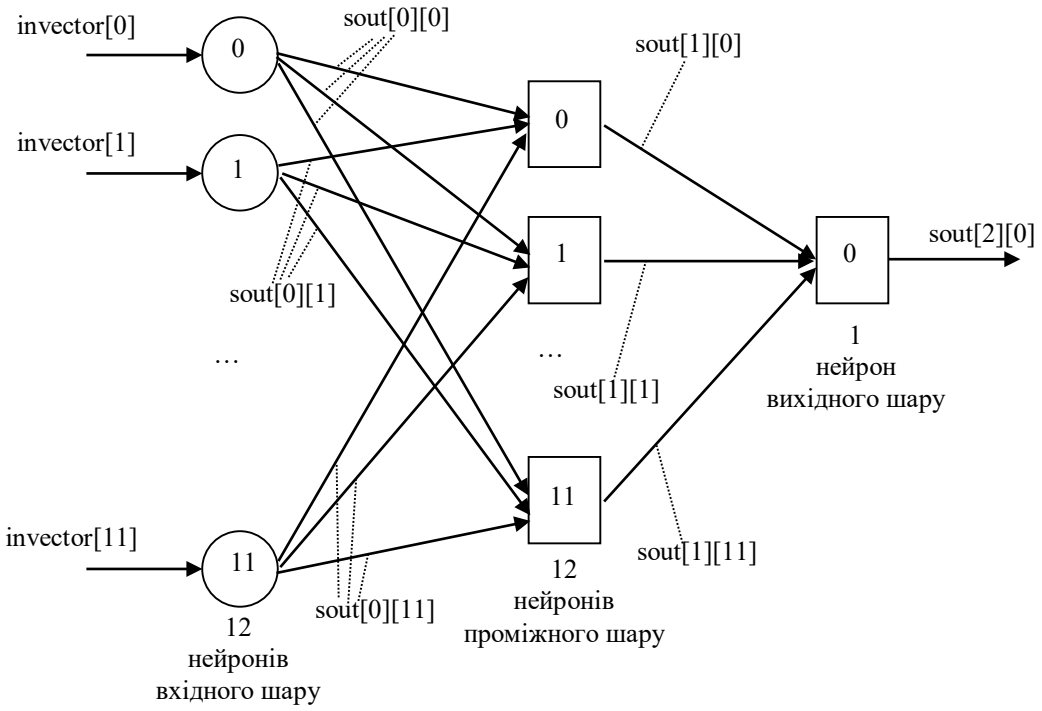


Рис. 3.5 Схема нейронної мережі, що розробляється,
із зазначенням змінних, які є виходами відповідних нейронів

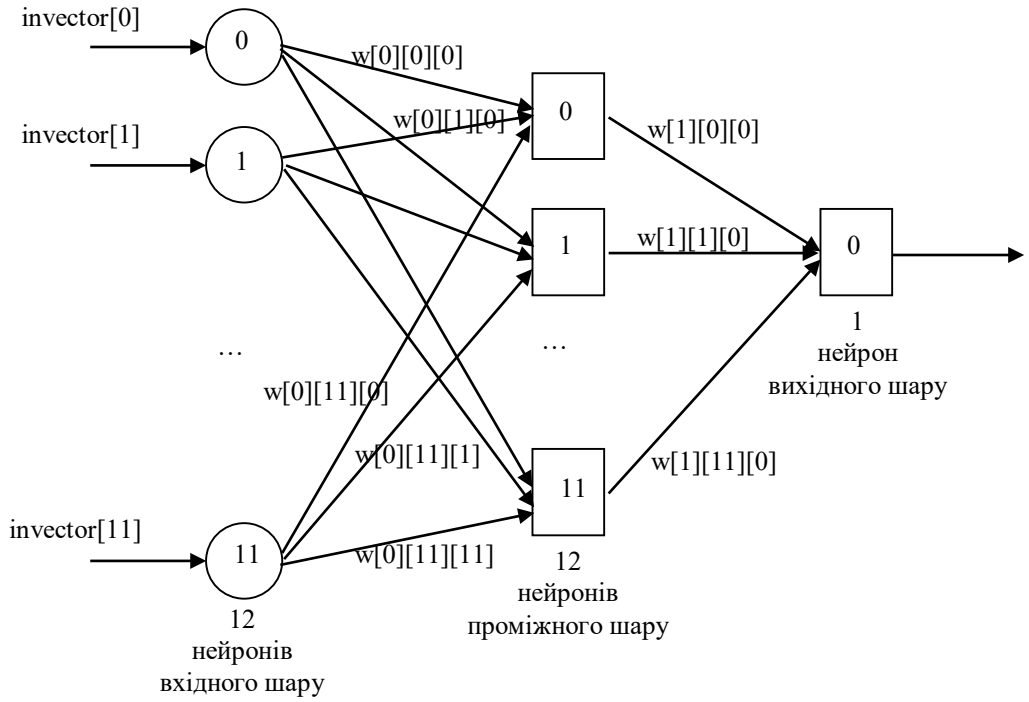


Рис. 3.6 Схема задання індексів у вагових коефіцієнтах,
що описують зв’язки між нейронами

З використанням сигналів sout між нейронами та ваг синаптичних зв'язків *w*, формуються зважені суми, які стають аргументами функції активації:

$$\text{sout[k+1][i]} = \text{Function}\left(\sum_{j=0}^{N_k} \text{sout[k][j]*w[k][j][i]}\right), \qquad (3.1)$$

де N_k – число нейронів у *k*-тому шарі, тобто Nneuronsinlayer[k];

Function – активаційна функція, яка задається як сигмоїдна і реалізує залежність:

$$f(x) = \frac{1}{1 + e^{-\alpha x}}, \qquad (3.2)$$

де α – параметр, за допомогою якого можна управляти швидкістю процесу навчання нейромережі; приймаємо $\alpha = 0{,}05$.

Залежності (3.2) відповідає наступна функція на мові C++:

```
float SigmoidalFunction(float arg)
{
    float res=(float)1/(1+exp(-alfa*arg));
    return res;
}
```

Переходячи до функцій, використаних у програмі, повний їх перелік наведений на рис. 3.7.

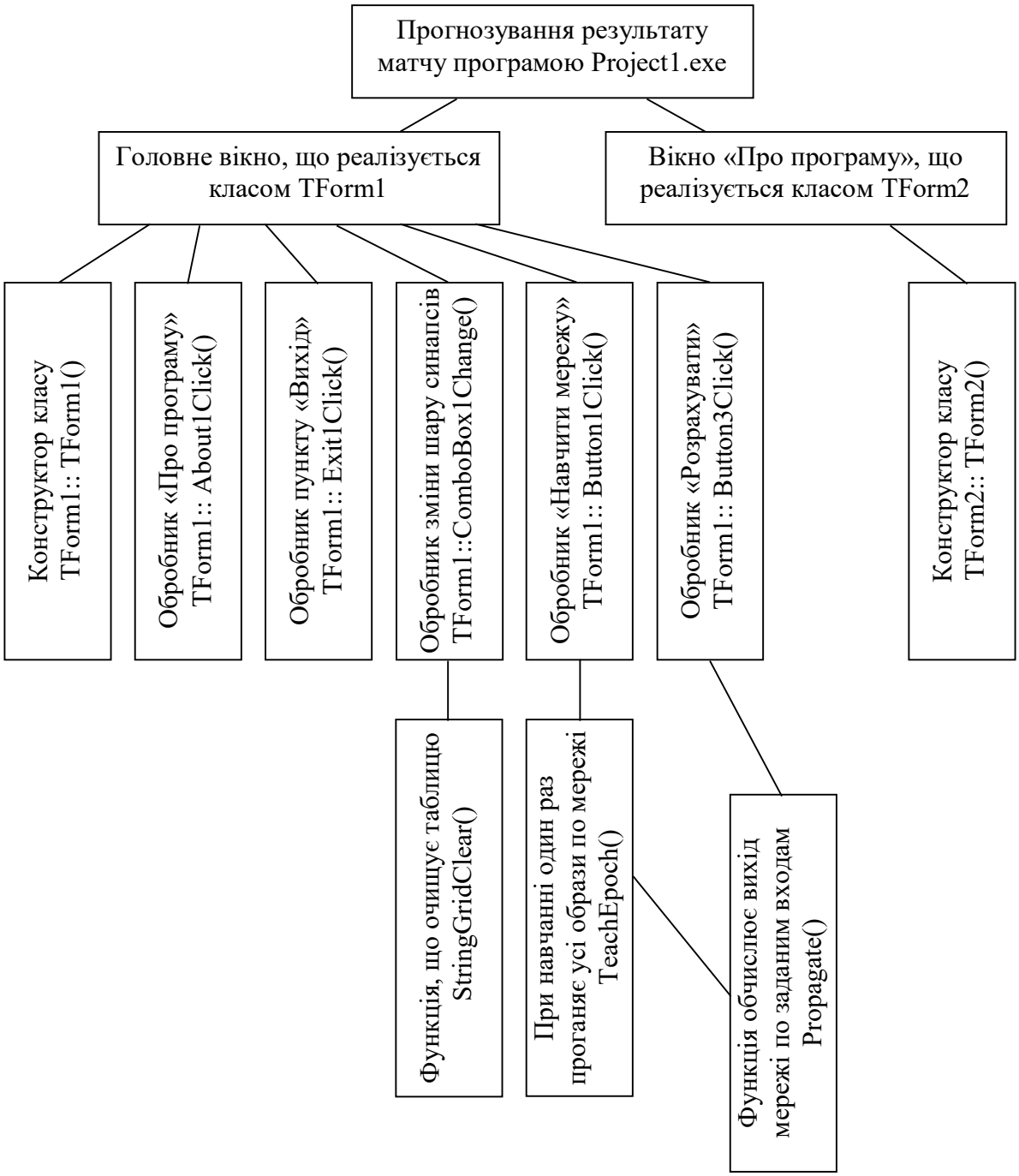


Рис. 3.7 Ієрархія функцій розробленої програми

Проведення вхідних сигналів, заданих масивом invector, через усю мережу виконується функцією Propagate():

```
void Propagate(float *inputarray)
{
```

```
        for(int i=0;i<Nneuronsinlayer[0];i++)
            sout[0][i]=(float)*(inputarray+
i*sizeof(float));
        float sum;
        for(int k=0;k<2;k++)
            for(int i=0;i<Nneuronsinlayer[k+1];i++)
            {
                sum=0;
                for(int j=0;j<Nneuronsinlayer[k];j++)
                    sum+=sout[k][j]*w[k][j][i];
                sout[k+1][i]=SigmoidalFunction(sum);
            }
    }
```

Аргументом цієї функції є стартова адреса inputarray масиву, в якому міститься вхідний вектор. Спочатку у функції зчитуються усі елементи цього float масиву (за допомогою звернення за адресою вказника `inputarray` та зміщення `i*sizeof(float)`) і формується перший набір сигналів `sout[0][i]`, що є виходами нейронів вхідного (нульового) шару.

Задається змінна sum, у якій буде зберігатися зважена сума вхідних сигналів для *i*-го нейрону. Вихідні сигнали формуються два рази: для прихованого шару та для вихідного шару. Кожного разу функція активації застосовується до зваженої суми усіх сигналів, що підходять до *i*-го нейрону. Таким чином, в кінці роботи функції Propagate() буде повністю сформованим набір вихідних сигналів для усіх нейронів (що зберігається в масиві sout).

Практичний інтерес представляє вихідний сигнал вихідного шару sout[2][0], який і являє собою шуканий результат матчу. Обробка та збереження єдиного вихідного значення реалізовані у масиві з одним елементом sout[2][0], що зроблено з метою уніфікації процесу розрахунку вихідних сигналів мережі, а

також з метою покращення масштабованості системи (у майбутньому, якщо буде необхідно, крім загального результату матчу прогнозувати, ще якісь характеристики, наприклад, результат першого сету).

Інші аспекти програмної реалізації не представляють особливої складності та зводяться до виконання відносно стандартних операцій переведення алгоритмічних складових у вихідні коди. З особливостями цих дій можна ознайомитися безпосередньо в коді продукту, наведеному у додатку до роботи.

3.4 Аналіз результатів тестування системи та оцінка адекватності її рішень

Після проведення реалізації програмного забезпечення обов'язковим етапом перед випуском програми в реліз є її тестування. Для створеної системи прогнозування результатів спортивних змагань з тенісу, в першу чергу, було проведено тривале тестування стабільності роботи, яке показало, що вона працює у штатному режимі, без виникнення системних помилок, непередбачених аварійних завершень тощо. Програма адекватно відпрацьовує різні нестандартні ситуації, наприклад, за відсутності файлу з даними для навчання видається інформаційне повідомлення (рис. 3.8).

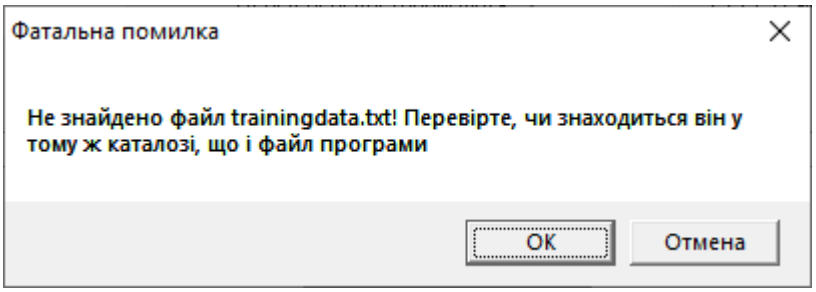


Рис. 3.8 Обробка виключень у програмі

Розроблена програма була протестована на адекватність виконання поставленої перед нею задачі – власне, прогнозування результатів матчів. Для

цього випадковим чином із наявної на початку досліджень сукупності усіх навчальних ситуацій (яких всього було зібрано 249 з [28]) було виключено 35, на яких здійснювалося не навчання, а перевірка правильності прогнозування навченої нейронної мережі.

Процедуру верифікації результату було автоматизовано: із файлу verifydata.txt зчитувалися 35 наявні там ситуації та для кожної перевірявся збіг реального результату матчу (записаний по системі 0 або 1, де 1 – виграш першого гравця, а 0 – виграш другого) із результатом, спрогнозованим системою. В кінці перевірки користувачу надається інформація про точність роботи системи (рис. 3.9).

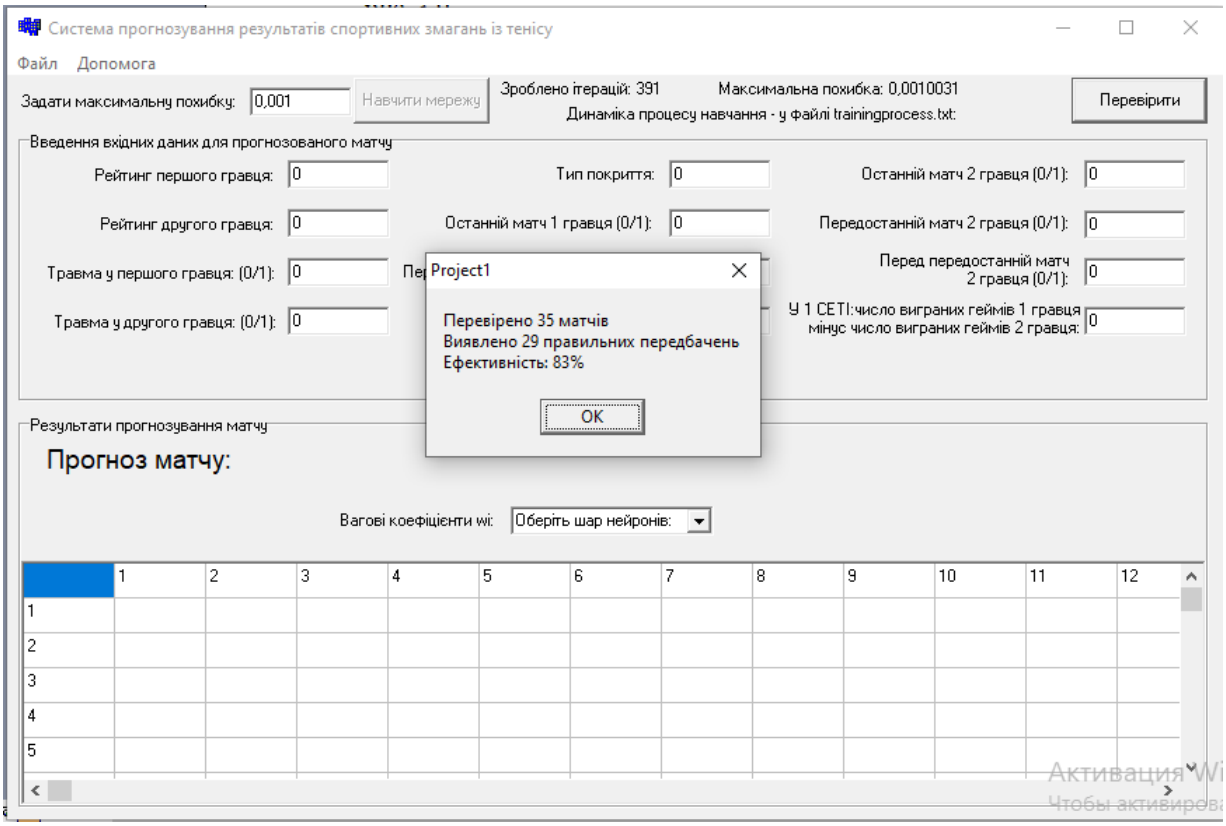


Рис. 3.9 Результат верифікації прогнозу

На рис. 3.9 наведено приклад результату використання системи при навчанні зі значенням максимальної похибки 0,001, яке задається за замовчуванням. При роботі з нейронною мережею потрібно перевірити вплив

цього параметру на ефективність її роботи, оскільки при істотному зменшенні похибки відбувається перенавчання мережі і прогнозування для ситуацій, що не входять у навчальну вибірку, стає значно менш ефективним. Відповідно, було виконано тестування з різними значеннями максимально допустимої похибки та проаналізовано, як при цьому змінюється ефективність роботи системи (табл. 3.1).

Таблиця 3.1 Залежність ефективності передбачень системи від заданої при навчанні максимальної похибки (дослідження перенавчання)

Задана максимальна похибка	Середня точність прогнозування, %
0,1	77
0,01	**83**
0,001	80
0,0001	76
0,00001	77

Можемо зробити висновок, що середнє значення похибки, яку дає система прогнозування, складає 17% (при заданій максимальній похибці рівній 0,01, але близькі результати дає і значення похибки 0,001 – 20% помилок). Задавати значення похибки менше 0,001 немає сенсу, оскільки при величинах порядку 0,00001 зростає час навчання, а також погіршується точність прогнозування випадків, що не входять у навчальну вибірку, тобто перенавчання системи.

Відповідно, розроблений програмний продукт має практичну цінність та може застосовуватися як при плануванні спортивної активності тенісистів, так і з метою оптимізації показника ROI (повернення інвестицій) при виконанні ставок в букмекерських організаціях.

3.5 Розробка комплекту документації для створеного програмного продукту

Після створення програми, що описується у попередньому підрозділі, потрібно розробити її документаційне забезпечення, і найважливіший документ при цьому – інструкція для користувача системи, основні елементи якого розглянемо у цьому підрозділі.

3.5.1 Інструкція для адміністратора по встановленню програми

Програмний продукт не потребує спеціальної процедури інсталяції, оскільки не використовує сторонніх бібліотек (навіть таких популярних на сьогоднішній день, як OpenCV, AForge тощо), а всю функціональність та процедури технічного характеру реалізує самостійно. Крім того, програма реалізована без необхідності її встановлення на цільовий ПК, а фактично є портативною версією.

Також немає потреби у встановленні та розгортанні окремої системи управління базами даних, оскільки вхідна інформація для роботи системи задається безпосередньо у її вікні, а інформацію для навчання зручно розмістити в окремому текстовому файлі.

Таким чином, весь процес встановлення системи на новий ПК полягає у копіюванні на локальний диск EXE-файлу програми у виділений користувачем для неї каталог.

3.5.2 Інструкція для користувача по використанню програми

В першу чергу потрібно зауважити, що у розробленого ПЗ є два режими роботи:

– навчання мережі;

– проведення прогнозування.

При кожному завантаженні програми спочатку потрібно провести її навчання, натиснувши кнопку «Навчити мережу».

Для правильного проведення цього процесу у тому ж каталозі, де знаходиться виконуваний файл програми, має бути розміщений файл datafortraining.txt із вибіркою даних для навчання мережі.

Структура файлу datafortraining.txt є наступною:

– спочатку через крапку з комою в одному рядку записуються значення показників у порядку, як вони приводяться у головному вікні програми;

– далі у цьому ж рядку через крапку з комою наводиться результат матчу, яким він був наприкінці реального змагання, причому «0» відповідає перемозі першого гравця, а «1» – перемозі другого;

– вказана структура рядка, що описується двома попередніми пунктами, повторюється стільки разів, скільки матчів є наявними у дата-сеті, що розглядається (тобто кожен рядок описує один матч, що відбувся, і який включений до навчальної вибірки).

Після натиснення на кнопку «Навчити мережу» потрібно дочекатися того, як ця кнопка стане неактивною, а кнопка «Розрахувати» – навпаки активується. Процес навчання мережі може здійснюватися від кількох секунд до кількох хвилин, що залежить від обчислювальної потужності процесора, на якому запускається програма, та від розміру навчальної вибірки.

Далі можливим стає використання і другого режиму роботи програми – прогнозування. Для цього потрібно внести у відповідні текстові поля значення характеристик матчу, результат якого буде прогнозуватися. Після цього можна натиснути кнопку «Розрахувати» і майже миттєво отримати необхідний результат.

За необхідності можна переглянути синаптичні ваги використовуваної нейронної мережі. Для цього у випадаючому списку «Ваги синапсів» потрібно обрати один із шарів і продивитися відповідні ваги у таблиці.

3.6 Висновки до розділу

У даному розділі проведено програмну реалізацію продукту, призначеного для прогнозування результатів спортивних змагань, а саме – великого тенісу. Обрано технології та засоби розробки:

– об’єктно-орієнтований підхід до програмування;

– C++ як зручну та функціональну мову загального призначення, що добре підходить для реалізації рішень із галузі інтелектуальних технологій;

– середовище розробки C++ Builder, яке надає найбільш зручний (простий та інтуїтивний) інтерфейс користувача, а також великий набір готових компонентів для вирішення багатьох функціональних задач «із коробки».

За допомогою обраних засобів та технологій розроблено проект інтерфейсу користувача, який включає засоби для введення вхідної інформації для роботи нейронної мережі, звичайно, результат прогнозування, а також елементи управління для відображення додаткової інформації, пов’язаної із використанням створеного ПЗ.

Також у розділі розглянуто особливості програмної реалізації продукту для прогнозування результатів спортивних змагань. Загалом, реалізація у повній мірі відповідає особливостям алгоритмічної основи, розробленої у попередньому розділі.

Для готового програмного продукту проведено тестування (системне, програмного продукту в цілому), яке показало належний рівень відповідності кінцевого результату поставленим у першому розділі вимогам. Програма працює без виникнення системних чи іншого виду помилок і адекватно виконує поставлену задачу.

Створено мінімальний комплект документації для розробленого програмного забезпечення (зокрема, інструкцію для користувача по роботі з системою).

ВИСНОВКИ

У даній роботі розроблено та реалізовано програмний продукт, який на базі методів машинного навчання здійснює прогнозування результатів спортивних змагань.

Як метод розв'язування обрана нейронна мережа у вигляді двошарового персептрону із 12 входами (в роботі обрано дванадцять основних характеристик майбутнього матчу, що можуть впливати на його результат) та одним виходом (з якого знімається прогнозований результат матчу у вигляді числа від 0 до 1, який можна трактувати як ступінь упевненості системи у перемозі першого гравця). Такий підхід дозволяє покращити процес прогнозування результатів спортивних змагань, використовуючи лише прості, але ефективні методи машинного навчання (нейронну мережу прямого поширення або персептрон). Головне – відпадає необхідність зведення класичної математичної моделі (яка, очевидно, була б досить складною, а також, насиченою різноманітними невідомими коефіцієнтами, які мають задаватися експертним шляхом, що часто буває не дуже надійним).

Програма написана на мові C++ у середовищі C++ Builder з використанням об'єктно-орієнтованого підходу до програмування. Тестування готового програмного продукту показало, що прогнозування здійснюється із задовільною для даної предметної галузі точністю 83 %. Відповідно, розроблений програмний продукт може використовуватися на практиці у плануванні турнірної активності спортсменів (що дозволяє оптимізувати її, наприклад, вкладаючи менше зусиль у матч, який майже достовірно буде програний, але приділяючи більше уваги матчам, у яких результат близький до 50:50), а також у букмекерській діяльності для оптимізації витрат учасників тоталізатору.

В подальших дослідженнях доцільно впровадити деякі елементи, що враховували б (очевидно, на рівні вірогідностей) рідкісні події, які тим не менше

можуть мати місце у сучасному світі (як, наприклад, нестандартні погодні умови, що можуть впливати на активність гравців на корті, або запровадження карантину та відсутність вболівальників на трибунах під час змагання, чого раніше ніколи не було). Із їх використанням результати прогнозування можуть бути покращені.

У цілому дослідження є завершеним, а мета роботи – досягнутою.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. T. Barnett and S. R. Clarke. Combining player statistics to predict outcomes of tennis matches. IMA Journal of Management Mathematics, 16:113-120, 2005.
2. J. A. O'Malley. Probability Formulas and Statistical Analysis in Tennis. Journal of Quantitative Analysis in Sports, 4(2), 2008.
3. W. J. Knottenbelt, D. Spanias, and A. M. Madurska. A common-opponent stochastic model for predicting the outcome of professional tennis matches. Computers and Mathematics with Applications, 64:3820-3827, 2012.
4. M. Madurska. A Set-By-Set Analysis Method for Predicting the Outcome of Professional Singles Tennis Matches. Technical report, Imperial College London, London, 2012.
5. S. R. Clarke and D. Dyte. Using official ratings to simulate major tennis tournaments. International Transactions in Operational Research, 7(6):585-594, 2000.
6. F. J. G. M. Klaassen and J. R. Magnus. Are Points in Tennis Independent and Identically Distributed? Evidence From a Dynamic Binary Panel Data Model. Journal of the American Statistical Association, 96:500-509, 2001.
7. Behter L.V., Klevets N.I. Weighted sum of indexes method for predicting football matches [Електронний ресурс] //Best.Today. – Режим доступу: http://bets.today/ru/articles/weighted-sum-of-indexes.
8. Douwe B. Predicting sports events from past results / B. Douwe // University of Twente. – 2011. – С. 1-6.
9. Igiri, Chinwe P. An improved prediction system for Football match result /P. Igiri, Chinwe, O. Nwachukwu, Enoch. //Department of Computer Science, University of Port Harcourt. – 2014. – С. 12-20.
10. Бєлка Д.О., Аверкина М.Ф. Моделі прогнозування футбольних матчів [Електронний ресурс]. – Режим доступу: https://naub.oa.edu.ua/2019/%d0%bc%d0%be%d0%b4%d0%b5%d0%bb%d1%96%d0%bf%d1%80%d0%be%d0%b3%d0%bd%d0%be%d0%b7%d1%83%d0%b2%d0%b

0%d0%bd%d0%bd%d1%8f%d1%84%d1%83%d1%82%d0%b1%d0%be%d0%bb%d1%8c%d0%bd%d0%b8%d1%85-%d0%bc%d0%b0%d1%82/

11. Семенюк В.О. Математичні моделі прогнозування результатів футбольних матчів [Текст] /В.О. Семенюк //XII Міжнародна науково-практична конференція «Інформаційні технології і автоматизація – 2019», Одеса, 17-18 жовтня 2019: збірник доповідей. – Одеса, 2019. – Ч.1. – С. 10-12.

12. Скороход А.В. Прогнозирование результатов спортивных событий на основе глубокой нейронной сети /А.В. Скороход //Міжнародний науковий журнал. – 2016. – № 7. – С. 122-123.

13. В.М. Кулик, Т.О. Коротєєва. Алгоритм прогнозування результатів футбольних матчів на основі нейронних мереж //Науковий вісник НЛТУ України, 2017, т. 27, № 9. – С.111-114.

14. Полухін О.А., Шаров С.В. Використання нейромережі для прогнозування результатів футбольних матчів //Інформаційні технології в освіті та науці: зб. наук. пр. №10, 2018 р. – С.214-217.

15. A. Somboonphokkaphan. Tennis Winner Prediction based on Time-Series History with Neural Modeling //IMECS 2009: International Multi-Conference of Engineers and Computer Scientists, Vol. I and II, I, 2009. – P.127-132.

16. Научные статьи в области искусственного интеллекта: Прогнозирование исходов спортивных игр методами нейросетевой кластеризации [Електронний ресурс] // Режим доступу: http://neuronus.com/stat/207

17. Штовба С.Д. Прогнозирование результатов футбольных матчей на основе нечетких правил /С.Д. Штовба, В.В. Вивдюк //Вестник молодых ученых. – 2002. – С. 57-64.

18. Заволодько А. Э. Прогнозирование результатов футбольных матчей на основе нечеткого многокритериального анализа /А.Э. Заволодько, М.И. Рыщенок

//Национальный технический университет «ХПИ». – 2009. – С. 129-131.

19. Методика прогнозирования с помощью теории нечетких множеств [Електронний ресурс]. – Режим доступу: http://rudbet.com/5-9-metodika-prognozirovaniya-s-pomoshhyu-teorii-nechetkih-mnozhestv/

20. Albin Y. Predicting outcomes of Soccer matches using machine learning / Y. Albina // Saint-Petersburg State University Mathematics and Mechanics Faculty. – 2014. – С. 3-12.

21. Kushal G. Football Match Winner Prediction / G. Kushal, S. Harshal, V. Saurabh, D. Khushali // Department of Computer Engineering, Dwarkadas J. Sanghvi College of Engineering, Mumbai, India. International Journal of Emerging Technology and Advanced Engineering, volume 5, issue 10, October 2015. – С. 364-368.

22. **Панюкова, Т. А. Языки и методы программирования. Создание простых GUI-приложений с помощью Visual C++. Учебное пособие /Т.А. Панюкова, А.В. Панюков. – Москва: Мир, 2015. – 744 с.**

23. **Полубенцева, М. C/C++. Процедурное программирование /М. Полубенцева. – М.: БХВ-Петербург, 2014. – 448 с.**

24. **Пахомов, Б. C/C++ и MS Visual C++ 2010 для начинающих /Б. Пахомов. – М.: БХВ-Петербург, 2011. – 736 с.**

25. **Зиборов, В. MS Visual C++ 2010 в среде .NET /В. Зиборов. – М.: Питер, 2012. – 320 с.**

26. **Довбуш, Галина. Visual C++ на примерах / Галина Довбуш, Анатолий Хомоненко. – М.: БХВ-Петербург, 2012. – 528 с.**

27. **Пахомов Б.И. Interbase и C++ Builder на примерах. – М.: БХВ-Петербург, 2006. – 288 с.**

28. **Sequential point-by-point data for tens of thousands of pro matches** [Електронний ресурс]. – Режим доступу: https://github.com/JeffSackmann/tennis_pointbypoint

ДОДАТОК А. ВИХІДНИЙ ТЕКСТ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОВОЮ С++

Файл Unit1.cpp:

```
#include <stdio.h>
#include <vcl.h>
#include <math.h>
#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"

#pragma package(smart_init)
#pragma resource "*.dfm"

const float alfa=0.05; //коефіцієнт для сигмоїдної математичної функції
const N=12; //кількість нейронів вхідного шару
const M=1;  //кількість нейронів вихідного шару
const T=1024; //максимальна кількість записів/ситуацій/образів для навчання мережі

TForm1 *Form1;
int Nneuronsinlayer[3]; //масив із кількостями нейронів у шарах: 12-12-1
int numtrainrecords=0; // реальна кількість записів для навчання мережі
float w[2][N][N];//Масив ваг синапсів. Індекси: номер шару, номер нейрона З ЯКОГО іде синапс, номер нейрона ДО ЯКОГО іде синапс
float sout[3][N];  //вихідні сигнали для усіх нейронів мережі
```

```
float invector[N],outvector[M]; //вхідний вектор, що подається на вхід мережі та вихідний, що знімається з виходу
float traindata[T][N+M]; // масив, у який із файлу зчитуються дані для навчання мережі
float delta[2][N]; //масив допоміжних дельт, які використовуються відповідно до алгоритму зворотного поширення помилки
float eta=1e-5; //коефіцієнт, що впливає на швидкість навчання
float prevres=0; //допоміжна змінна для перевірки чи зменшуються постійно помилки при навчанні мережі (або процес іде хаотично)
float
maxvals[]={50,3000,1000,100000,100,5000,5000,12,10000,1500,100000};
//масив максимальних значень усіх вхідних величин, що дозволяє шляхом ділення на його елементи зблизити усі їх значення (нормалізувати їх), інакше дуже важко навчити систему, якщо рівноправні по суті входи отримують значення, які відрізняються на кілька порядків
//---------------------------------------------------------------

float SigmoidalFunction(float arg) //математична сигмоїдна функція Для обчислення виходу нейрона
{
    float res=(float)1/(1+exp(-alfa*arg));
    return res;
}

void StringGridClear(TStringGrid* tsg) //функція, яка очищує елемент-таблицю ваг StringGrid, але не повністю, залишаються верхній і лівий ряди
{
    for(int i=1;i<=tsg->ColCount;i++)
        for(int j=1;j<=tsg->RowCount;j++)
            tsg->Cells[i][j]="";
}
```

```
__fastcall TForm1::TForm1(TComponent* Owner) //конструктор головного вікна
    : TForm(Owner)
{
    Nneuronsinlayer[0]=N;  //задаються кількості нейронів у шарах в цьому масиві, такий масив використовується для зручності роботи з циклами
    Nneuronsinlayer[1]=N;
    Nneuronsinlayer[2]=M;
    for(int i=1;i<=N;i++)   //цикл заповнення заголовків таблиці ваг
    {
        StringGrid1->Cells[0][i]=i;
        StringGrid1->Cells[i][0]=i;
    }
    srand(time(0)); //стандартна ініціалізація генератора випадкових чисел, аби кожного разу генерувалися нові початкові ваги синапсів, адже вони спочатку задаються випадковими числами
    for(int  k=0;k<2;k++) //якраз цикли задання ваг усіх синапсів випадковими значеннями від -1 до +1
        for(int i=0;i<Nneuronsinlayer[k];i++)
            for(int j=0;j<Nneuronsinlayer[k+1];j++)
                w[k][i][j]=(float)rand()/RAND_MAX*pow(-1,rand());
}
//---------------------------------------------------------------

void __fastcall TForm1::About1Click(TObject *Sender)
{
    Form2->ShowModal();
}
//---------------------------------------------------------------
void __fastcall TForm1::Exit1Click(TObject *Sender)
{
    exit(0);
```

```
}
//---------------------------------------------------------------
void __fastcall TForm1::ComboBox1Change(TObject *Sender) //функція що змінює таблицю ваг синапсів, відповідно до того, який шар обрав користувач. По суті тут дві опції: ваги синапсів від входу до прихованого шару та синапсів від прихованого шару до виходу
{
    StringGridClear(StringGrid1);
    int k=ComboBox1->ItemIndex;
    if(k==0)
        StringGrid1->RowCount=N+1;
    if(k==1)
        StringGrid1->RowCount=M+1;
    for(int i=0;i<Nneuronsinlayer[k];i++)
        for(int j=0;j<Nneuronsinlayer[k+1];j++)
            StringGrid1-
>Cells[i+1][j+1]=FloatToStrF(w[k][i][j],ffFixed,5,5);
}
//функція розповсюдження вхідних сигналів по всій мережі і в результаті отримання вихідних сигналів:
void Propagate(float *inputarray)
{
//формування виходів нейронів вхідного шару, шляхом простого повторення, це допоміжний шар:
    for(int i=0;i<Nneuronsinlayer[0];i++)
        sout[0][i]=(float)*(inputarray+i*sizeof(float));
    float sum;
//цикл, що виконується для виходів усіх шарів:
    for(int k=0;k<2;k++) //k - номер шару нейронів
//цикл, що виконується для усіх нейронів, слідуючих за k-им шаром
        for(int i=0;i<Nneuronsinlayer[k+1];i++)
        {
            sum=0;
```

```
            for(int j=0;j<Nneuronsinlayer[k];j++)
                sum+=sout[k][j]*w[k][j][i];
            sout[k+1][i]=SigmoidalFunction(sum);
        }
}

float TeachEpoch()
{
    float error=0;
    for(int k=0;k<numtrainrecords;k++)
    {
        for(int i=0;i<Nneuronsinlayer[0];i++)
        {
            invector[i]=traindata[k][i];
        }
        for(int i=0;i<Nneuronsinlayer[2];i++)
        {
            outvector[i]=traindata[k][N+i];
        }
        Propagate(invector);
        for(int i=0;i<Nneuronsinlayer[2];i++)
        {
            for(int j=0;j<Nneuronsinlayer[1];j++)
            {
                delta[1][i]=sout[2][i]*(1-
sout[2][i])*(outvector[i]-sout[2][i]);
                w[1][j][i]+=eta*delta[1][i]*sout[1][j];
            }
        }
        for(int i=0;i<Nneuronsinlayer[1];i++)
        {
            float sum1=0;
            for(int m=0;m<Nneuronsinlayer[2];m++)
```

```
                sum1+=delta[1][m]*w[1][i][m];
            for(int j=0;j<Nneuronsinlayer[0];j++)
            {
                delta[0][i]=sout[1][i]*(1-sout[1][i])*sum1;
                w[0][j][i]+=eta*delta[0][i]*sout[0][j];
            }
        }
        for(int i=0;i<Nneuronsinlayer[2];i++)
        {
            if(fabs(outvector[i]-sout[2][i])>error)
                error=fabs(outvector[i]-sout[2][i]);
        }
    }
    return error;
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
    float res=0,minres=1,dirres=0;
    int i=0,k=0;
    eta=1;
    Label8->Caption="Training...";
    //відкриваємо файл trainingdata.txt
    FILE *f;
    char str[255],*str1,*next, word[8];
    try
    {
      f=fopen("trainingdata.txt", "r");
      i=0;
      while(!feof(f))  //поки не закінчиться файл зчитуємо по одному рядку і розбиваємо його символом ";" на окремі слова, кожне з яких перетворюємо на дробове число
      {
```

```
          fgets((char*)str, 255, f);
        str1=str;

        for(k=0;k<N;k++)
        {
            next=strchr(str1,';');
            int v=next-str1;
            strncpy(word,str1,next-str1);
            word[next-str1]=0;

traindata[i][k]=StrToFloat(Trim((AnsiString)word))/maxvals[k];
            str1=next+1;
        }
        next=strchr(str1,'\n');
        int v=next-str1;
        strncpy(word,str1,next-str1);
        word[next-str1]=0;

traindata[i][k]=StrToFloat(Trim((AnsiString)word))/maxvals[k];   //
запис вихідного сигналу
        i++;
      }
      fclose(f); //все зчитали і закрили файл
    }
    catch(...)  //обробка виключення, яке виникає, коли файл
trainingdata.txt не існує
    {
      Application->MessageBox("Не знайдено файл trainingdata.txt!
Перевірте, чи знаходиться він у тому ж каталозі, що і файл програми",
"Фатальна помилка", IDOK);
    }
    numtrainrecords=i;
```

```
    do { //Безпосередньо цикл навчання мережі
        Label5->Caption="Зроблено ітерацій: "+IntToStr(i);
        Label6->Caption="Максимальна                похибка:
"+FloatToStrF(res,ffFixed,3,3);
        res=TeachEpoch();//виконуємо один раунд навчання по усім
образам і отримуємо максимальну помилку у відносних одиницях
        if(res<minres)
            minres=res; //це мінімальна помилка за усі раунди
навчання
        if(res>=prevres)
            dirres++; //при навчанні методом зворотного поширення
помилки теоретично мережа може піти по неправильному шляху і замість
зменшення помилки вона систематично почне збільшуватися. Аби уникнути
такої ситуації і зациклювання процесу навчання введена змінна dirres
(напрямок зміни результату) і якщо вона більше 1000 разів говорить
нам, що похибка зростає а не зменшується, то треба вийти із циклу.
Взагалі така ситуація буває рідко і майже завжди вихід відбувається
по помилці менше 0,07
        else
            dirres=0;
        i++;
        prevres=res;

    } while((res>0.07)&&(dirres<1000)); //виконуємо навчання, поки
на усіх наборах вхідних даних не буде помилка менше 7%
    Label8->Caption="";
    Button3->Enabled=true;
    Button1->Enabled=false;
    TrainNetwork1->Enabled=false;
    Button1->Hint="Мережа вже навчена";
}
//---------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
```

```
{
//Усі введені у текстові поля вхідні дані нормалізуються діленням на
відповідний елемент масиву maxvals та записується у вхідний вектор
    invector[0]=StrToFloat(Edit1->Text)/maxvals[0];
    invector[1]=StrToFloat(Edit2->Text)/maxvals[1];
    invector[2]=StrToFloat(Edit3->Text)/maxvals[2];
    invector[3]=StrToFloat(Edit4->Text)/maxvals[3];
    invector[4]=StrToFloat(Edit5->Text)/maxvals[4];
    invector[5]=StrToFloat(Edit6->Text)/maxvals[5];
    invector[6]=StrToFloat(Edit7->Text)/maxvals[6];
    invector[7]=StrToFloat(Edit8->Text)/maxvals[7];
    invector[8]=StrToFloat(Edit9->Text)/maxvals[8];
    invector[9]=StrToFloat(Edit10->Text)/maxvals[9];
    invector[10]=StrToFloat(Edit11->Text)/maxvals[10];
    invector[11]=StrToFloat(Edit12->Text)/maxvals[11];
//проведення вхідних даних по усій мережі:
    Propagate(invector);
//записати результат у відповідне поле у головному вікні:
    Label8-
>Caption=FloatToStrF(sout[2][0]*maxvals[10],ffFixed,5,0);
}
```

Файл Unit2.cpp:

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit2.h"
//---------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
    : TForm(Owner)
```

```
{
}
//---------------------------------------------------------------

void __fastcall TForm2::Button1Click(TObject *Sender)
{
        Close();
}
//---------------------------------------------------------------
```

Файл Project1.cpp:

```
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------
USEFORM("Unit1.cpp", Form1);
USEFORM("Unit2.cpp", Form2);
//---------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
    try
    {
         Application->Initialize();
         Application->CreateForm(__classid(TForm1), &Form1);
         Application->CreateForm(__classid(TForm2), &Form2);
         Application->Run();
    }
    catch (Exception &exception)
    {
         Application->ShowException(&exception);
    }
    catch (...)
    {
```

```
        try
        {
            throw Exception("");
        }
        catch (Exception &exception)
        {
            Application->ShowException(&exception);
        }
    }
    return 0;
}
//---------------------------------------------------------------------
```

Файл Unit1.dfm:

```
object Form1: TForm1
  Left = 306
  Top = 120
  Width = 883
  Height = 590
  Caption          =          #1057#1080#1089#1090#1077#1084#1072'
'#1087#1088#1086#1075#1085#1086#1079#1091#1074#1072#1085#1085#1103'
'#1088#1077#1079#1091#1083#1100#1090#1072#1090#1110#1074'
'#1089#1087#1086#1088#1090#1080#1074#1085#1080#1093'
'#1079#1084#1072#1075#1072#1085#1100'                    '#1110#1079'
'#1090#1077#1085#1110#1089#1091
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  Menu = MainMenu1
```

```
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object Label19: TLabel
    Left = 488
    Top = 12
    Width = 94
    Height = 13
    Hint =
      'Number of training epochs carried out while training Neural
Netw' +
      'ork'
    Caption       =       #1047#1088#1086#1073#1083#1077#1085#1086'
'#1110#1090#1077#1088#1072#1094#1110#1081':'
    ParentShowHint = False
    ShowHint = True
  end
  object Label18: TLabel
    Left = 664
    Top = 12
    Width = 118
    Height = 13
    Hint = 'Maximum error value which was achieved while training'
    Caption                                                    =
#1052#1072#1082#1089#1080#1084#1072#1083#1100#1085#1072'
'#1087#1086#1093#1080#1073#1082#1072':'
    ParentShowHint = False
    ShowHint = True
  end
  object Button1: TButton
    Left = 378
    Top = 0
    Width = 97
```

```
    Height = 33
    Hint =
      'Press this button firstly and only one time. Then wait some
minu' +
      'tes'
    Caption          =          #1053#1072#1074#1095#1080#1090#1080'
'#1084#1077#1088#1077#1078#1091
    ParentShowHint = False
    ShowHint = True
    TabOrder = 0
    OnClick = Button1Click
  end
  object GroupBox1: TGroupBox
    Left = 8
    Top = 39
    Width = 849
    Height = 193
    Caption       =       #1042#1074#1077#1076#1077#1085#1085#1103'
'#1074#1093#1110#1076#1085#1080#1093'   '#1076#1072#1085#1080#1093'
'#1076#1083#1103'
'#1087#1088#1086#1075#1085#1086#1079#1086#1074#1072#1085#1086#1075#
1086' '#1084#1072#1090#1095#1091
    TabOrder = 1
    object Label1: TLabel
      Left = 54
      Top = 24
      Width = 128
      Height = 13
      Caption        =        #1056#1077#1081#1090#1080#1085#1075'
'#1087#1077#1088#1096#1086#1075#1086'
'#1075#1088#1072#1074#1094#1103':'
    end
    object Label2: TLabel
```

```
      Left = 58
      Top = 59
      Width = 124
      Height = 13
      Caption        =        #1056#1077#1081#1090#1080#1085#1075'
'#1076#1088#1091#1075#1086#1075#1086'
'#1075#1088#1072#1074#1094#1103':'
    end
    object Label3: TLabel
      Left = 20
      Top = 94
      Width = 163
      Height = 13
      Caption     =     #1058#1088#1072#1074#1084#1072'     '#1091'
'#1087#1077#1088#1096#1086#1075#1086'
'#1075#1088#1072#1074#1094#1103': (0/1):'
    end
    object Label10: TLabel
      Left = 25
      Top = 129
      Width = 159
      Height = 13
      Caption     =     #1058#1088#1072#1074#1084#1072'     '#1091'
'#1076#1088#1091#1075#1086#1075#1086'
'#1075#1088#1072#1074#1094#1103': (0/1):'
    end
    object Label12: TLabel
      Left = 383
      Top = 23
      Width = 71
      Height = 13
      Caption                  =                  #1058#1080#1087'
'#1087#1086#1082#1088#1080#1090#1090#1103':'
```

```
    end
    object Label13: TLabel
      Left = 304
      Top = 58
      Width = 148
      Height = 21
      Alignment = taRightJustify
      Caption      =      #1054#1089#1090#1072#1085#1085#1110#1081'
'#1084#1072#1090#1095' 1 '#1075#1088#1072#1074#1094#1103' (0/1):'
      WordWrap = True
    end
    object Label20: TLabel
      Left = 272
      Top = 93
      Width = 180
      Height = 19
      Alignment = taRightJustify
      Caption                                                    =
#1055#1077#1088#1077#1076#1086#1089#1090#1072#1085#1085#1110#1081'
'#1084#1072#1090#1095' 1 '#1075#1088#1072#1074#1094#1103' (0/1):'
      WordWrap = True
    end
    object Label14: TLabel
      Left = 312
      Top = 120
      Width = 140
      Height = 33
      Alignment = taRightJustify
      Caption             =             #1055#1077#1088#1077#1076'
'#1087#1077#1088#1077#1076#1086#1089#1090#1072#1085#1085#1110#1081'
'#1084#1072#1090#1095' 1 '#1075#1088#1072#1074#1094#1103' (0/1):'
      WordWrap = True
    end
```

```
    object Label9: TLabel
      Left = 601
      Top = 23
      Width = 148
      Height = 13
      Alignment = taRightJustify
      Caption      =      #1054#1089#1090#1072#1085#1085#1110#1081'
'#1084#1072#1090#1095' 2 '#1075#1088#1072#1074#1094#1103' (0/1):'
      WordWrap = True
    end
    object Label11: TLabel
      Left = 571
      Top = 58
      Width = 178
      Height = 13
      Alignment = taRightJustify
      Caption                                                    =
#1055#1077#1088#1077#1076#1086#1089#1090#1072#1085#1085#1110#1081'
'#1084#1072#1090#1095' 2 '#1075#1088#1072#1074#1094#1103' (0/1):'
      WordWrap = True
    end
    object Label15: TLabel
      Left = 611
      Top = 85
      Width = 138
      Height = 26
      Alignment = taRightJustify
      Caption             =             #1055#1077#1088#1077#1076'
'#1087#1077#1088#1077#1076#1086#1089#1090#1072#1085#1085#1110#1081'
'#1084#1072#1090#1095' 2 '#1075#1088#1072#1074#1094#1103' (0/1):'
      WordWrap = True
    end
    object Label16: TLabel
```

```
      Left = 544
      Top = 120
      Width = 213
      Height = 39
      Alignment = taRightJustify
      Caption =
        #1059' 1 '#1057#1045#1058#1030':'#1095#1080#1089#1083#1086'
'#1074#1080#1075#1088#1072#1085#1080#1093'
'#1075#1077#1081#1084#1110#1074' 1 '#1075#1088#1072#1074#1094#1103'
'#1084#1110#1085#1091#1089'             '#1095#1080#1089#1083#1086'
'#1074#1080#1075#1088#1072#1085#1080#1093' '#1075#1077#1081 +
        #1084#1110#1074' 2 '#1075#1088#1072#1074#1094#1103':'
      WordWrap = True
    end
    object Edit1: TEdit
      Left = 192
      Top = 20
      Width = 73
      Height = 21
      TabOrder = 0
      Text = '0'
    end
    object Edit2: TEdit
      Left = 192
      Top = 55
      Width = 73
      Height = 21
      TabOrder = 1
      Text = '0'
    end
    object Edit3: TEdit
      Left = 192
      Top = 90
```

```
  Width = 73
  Height = 21
  TabOrder = 2
  Text = '0'
end
object Edit4: TEdit
  Left = 192
  Top = 125
  Width = 73
  Height = 21
  TabOrder = 3
  Text = '0'
end
object Edit6: TEdit
  Left = 464
  Top = 20
  Width = 73
  Height = 21
  TabOrder = 4
  Text = '0'
end
object Edit7: TEdit
  Left = 464
  Top = 55
  Width = 73
  Height = 21
  TabOrder = 5
  Text = '0'
end
object Edit8: TEdit
  Left = 464
  Top = 90
  Width = 73
```

```
  Height = 21
  TabOrder = 6
  Text = '0'
end
object Edit9: TEdit
  Left = 464
  Top = 125
  Width = 73
  Height = 21
  TabOrder = 7
  Text = '0'
end
object Edit10: TEdit
  Left = 760
  Top = 20
  Width = 73
  Height = 21
  TabOrder = 8
  Text = '0'
end
object Edit5: TEdit
  Left = 760
  Top = 56
  Width = 73
  Height = 21
  TabOrder = 9
  Text = '0'
end
object Edit11: TEdit
  Left = 760
  Top = 90
  Width = 73
  Height = 21
```

```
      TabOrder = 10
      Text = '0'
    end
    object Edit12: TEdit
      Left = 760
      Top = 125
      Width = 73
      Height = 21
      TabOrder = 11
      Text = '0'
    end
    object Button3: TButton
      Left = 370
      Top = 153
      Width = 97
      Height = 33
      Hint = 'Press this button to recognize'
      Caption                                                    =
#1056#1086#1079#1088#1072#1093#1091#1074#1072#1090#1080
      Enabled = False
      ParentShowHint = False
      ShowHint = True
      TabOrder = 12
      OnClick = Button3Click
    end
  end
  object GroupBox2: TGroupBox
    Left = 8
    Top = 242
    Width = 849
    Height = 281
```

```
    Caption  =  #1056#1077#1079#1091#1083#1100#1090#1072#1090#1080'
'#1087#1088#1086#1075#1085#1086#1079#1091#1074#1072#1085#1085#1103'
'#1084#1072#1090#1095#1091
    TabOrder = 2
    object Label7: TLabel
      Left = 20
      Top = 20
      Width = 132
      Height = 22
      Hint = 'Here you can see values on outputs of neural network'
      Caption        =        #1055#1088#1086#1075#1085#1086#1079'
'#1084#1072#1090#1095#1091':'
      Font.Charset = DEFAULT_CHARSET
      Font.Color = clWindowText
      Font.Height = -19
      Font.Name = 'Arial'
      Font.Style = []
      ParentFont = False
      ParentShowHint = False
      ShowHint = True
    end
    object Label8: TLabel
      Left = 204
      Top = 20
      Width = 571
      Height = 24
      Caption      =      #1057#1087#1086#1095#1072#1090#1082#1091'
'#1085#1072#1074#1095#1110#1090#1100'
'#1084#1077#1088#1077#1078#1091'!                    '#1062#1077'
'#1084#1086#1078#1077'             '#1079#1072#1085#1103#1090#1080'
'#1076#1077#1082#1110#1083#1100#1082#1072'
'#1093#1074#1080#1083#1080#1085
      Font.Charset = RUSSIAN_CHARSET
```

```
      Font.Color = clWindowText
      Font.Height = -21
      Font.Name = 'Arial'
      Font.Style = []
      ParentFont = False
    end
    object Label4: TLabel
      Left = 229
      Top = 68
      Width = 110
      Height = 13
      Caption           =           #1042#1072#1075#1086#1074#1110'
'#1082#1086#1077#1092#1110#1094#1110#1108#1085#1090#1080' wi:'
    end
    object Label6: TLabel
      Left = 264
      Top = 236
      Width = 118
      Height = 13
      Hint = 'Maximum error value which was achieved while training'
      Caption                                                    =
#1052#1072#1082#1089#1080#1084#1072#1083#1100#1085#1072'
'#1087#1086#1093#1080#1073#1082#1072':'
      ParentShowHint = False
      ShowHint = True
    end
    object Label5: TLabel
      Left = 264
      Top = 213
      Width = 94
      Height = 13
      Hint =
```

```
        'Number of training epochs carried out while training Neural
Netw' +
        'ork'
      Caption      =      #1047#1088#1086#1073#1083#1077#1085#1086'
'#1110#1090#1077#1088#1072#1094#1110#1081':'
      ParentShowHint = False
      ShowHint = True
    end
    object ComboBox1: TComboBox
      Left = 351
      Top = 65
      Width = 145
      Height = 21
      Hint =
        'There are two sets of connections  between neurons: from
input ' +
        'layer to hidden, and from hidden layer to output, so choose
one ' +
        'of two layers'
      ItemHeight = 13
      ParentShowHint = False
      ShowHint = True
      TabOrder = 0
      Text = #1054#1073#1077#1088#1110#1090#1100' '#1096#1072#1088'
'#1085#1077#1081#1088#1086#1085#1110#1074':'
      OnChange = ComboBox1Change
      Items.Strings = (
        'Input'
        'Hidden')
    end
    object StringGrid1: TStringGrid
      Left = 2
      Top = 104
```

```
      Width = 845
      Height = 175
      Hint =
        'Here are weights of synapses from one to another neuron.
Connect' +
        'ion from HorisontalNumber to VerticalNumber'
      Align = alBottom
      ColCount = 27
      FixedCols = 0
      RowCount = 64
      FixedRows = 0
      ParentShowHint = False
      ShowHint = True
      TabOrder = 1
      RowHeights = (
        24
...
        24)
    end
  end
  object MainMenu1: TMainMenu
    Left = 808
    Top = 65528
    object File1: TMenuItem
      Caption = #1060#1072#1081#1083
      object TrainNetwork1: TMenuItem
        Caption       =       #1053#1072#1074#1095#1080#1090#1080'
'#1084#1077#1088#1077#1078#1091
        OnClick = Button1Click
      end
      object Open1: TMenuItem
```

```
        Caption                                                    =
#1056#1086#1079#1088#1072#1093#1091#1074#1072#1090#1080'
'#1088#1077#1079#1091#1083#1100#1090#1072#1090
      end
      object Exit1: TMenuItem
        Caption = #1042#1080#1093#1110#1076
        OnClick = Exit1Click
      end
    end
    object Help1: TMenuItem
      Caption = #1044#1086#1087#1086#1084#1086#1075#1072
      object About1: TMenuItem
        Caption                  =                  #1055#1088#1086'
'#1087#1088#1086#1075#1088#1072#1084#1091'...'
        OnClick = About1Click
      end
    end
  end
end
```

Файл Unit2.dfm:

```
object Form2: TForm2
  Left = 606
  Top = 263
  BorderIcons = [biSystemMenu]
  BorderStyle = bsDialog
  Caption                    =                    #1055#1088#1086'
'#1087#1088#1086#1075#1088#1072#1084#1091':'
  ClientHeight = 138
  ClientWidth = 268
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
```

```
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object Label1: TLabel
    Left = 96
    Top = 24
    Width = 161
    Height = 80
    Caption =
      #1055#1088#1086#1075#1088#1072#1084#1072'
'#1087#1088#1086#1075#1085#1086#1079#1091#1074#1072#1085#1085#1103'
'#1088#1077#1079#1091#1083#1100#1090#1072#1090#1110#1074'
'#1089#1087#1086#1088#1090#1080#1074#1085#1080#1093'
'#1079#1084#1072#1075#1072#1085#1100'                    '#1079'
'#1090#1077#1085#1110#1089#1091', ' +
      '2021.'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -13
    Font.Name = 'MS Sans Serif'
    Font.Style = []
    ParentFont = False
    WordWrap = True
  end
  object Image1: TImage
    Left = 16
    Top = 24
    Width = 73
    Height = 65
```

```
    Picture.Data = {

0A544A504547496D61676558750000FFD8FFE000104A46494600010101000000
...
      9452D1401FFFD9}
    Proportional = True
  end
  object Button1: TButton
    Left = 96
    Top = 104
    Width = 75
    Height = 25
    Caption = #1054#1050
    TabOrder = 0
    OnClick = Button1Click
  end
end
```