Міністерство освіти і науки, молоді та спорту України
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

До захисту допущено
Завідувач кафедри
комп'ютерної інженерії
к.т.н., доц. О.М.Березький
_____
"___" _____ 20___ р.

ДИПЛОМНА РОБОТА
освітньо-кваліфікаційного рівня "Магістр"
зі спеціальності 8.05010201 "Комп'ютерні системи та мережі"
на тему:

НЕЙРОМЕРЕЖЕВІ МОДЕЛІ ПОВЕДІНКИ КОРИСТУВАЧІВ КОМП'ЮТЕРНИХ СИСТЕМ

Студент групи КСМм - 51
Малюга А.М.

_____
підпис

Науковий керівник
к.т.н., доцент Палій І.О.

_____
підпис

Консультант з нормоконтролю
Палій І.О.

_____
Прізвище, ініціали Підпис

Тернопіль – 2012

Міністерство освіти і науки, молоді та спорту України
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

"Затверджую"
Зав. кафедри
комп'ютерної інженерії
к.т.н., доц. О.М. Березький
_____
"____"_____20__ р.

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТА

Малюги Андрія Михайловича

1. **Тема дипломної роботи** "Нейромережеві моделі поведінки користувачів комп'ютерних систем" затверджена наказом університету №_____ від „____" _____20__ р

2.**Термін здачі** закінченої дипломної роботи _____

3. **Об'єкт дослідження**: процеси людино-машинної взаємодії.

4. **Предмет дослідження**: нейромережеві моделі аналізу поведінки користувачів комп'ютерних систем.

5. **Перелік задач, які мають бути вирішені**:
- дослідити і класифікувати існуючі методи і підходи до побудови моделей поведінки користувачів комп'ютерних систем;
- розробити комплексну модель користувачів комп'ютерних систем, яка враховуватиме як динамічні, такі статистичні властивості їх поведінки, а також можливі зміни в їх поведінці;
- розробити генеративну модель поведінки користувачів;
- забезпечити верифікацію розробленої моделі на реальних даних;
- розробити і реалізувати систему моніторингу діяльності користувачів комп'ютерних систем в корпоративних мережах з використанням об'єктно-орієнтованого підходу і агентної технології.

6. **Перелік ілюстративного матеріалу:**
- структура комплексної моделі поведінки користувача,
- приклад архітектури штучної нейронної мережі,
- структура інтерактивної моделі,
- структура сеансової моделі,
- структура комплексної моделі,
- архітектура системи моніторингу,
- UML–діаграми,
- графічний інтерфейс користувача.

7.**Консультанти по роботі**

Розділ	Консультант	Підпис
1		
2		
3		

КАЛЕНДАРНИЙ ПЛАН

№	Назва структурних частин ДР	Термін виконання	Примітка
1	Аналіз методів побудови і областей застосування моделей поведінки користувачів	15.09.2011 – 5.11.2011	
2	Нейромережеві моделі поведінки користувачів комп'ютерних систем	6.11.2011 – 31.01.2012	
3	Реалізація нейромережевої моделі поведінки користувачів комп'ютерних систем на основі агентної технології	1.02.2012 – 23.04.2012	

Завдання прийняв до виконання _____
(підпис)

Керівник дипломної роботи _____
(підпис)

РЕФЕРАТ

Дипломна робота на тему “Нейромережеві моделі поведінки користувачів комп’ютерних систем” на здобуття освітньо-кваліфікаційного рівня “Магістр” зі спеціальності “Комп’ютерні системи та мережі” написана обсягом 86 сторінок і містить 16 ілюстрацій, 2 додатки та 58 джерел за переліком посилань.

Метою роботи є розробка ефективних моделей аналізу поведінки користувачів комп'ютерних систем і їх застосування в системах моніторингу і виявлення аномальної діяльності.

Методи досліджень. В дипломній роботі використовуються методи системного аналізу складних взаємозв'язаних процесів, нейромережевий підхід, статистичне моделювання, об'єктно-орієнтований підхід до моделювання і розробки складних систем; агентна парадигма, методи чисельного експерименту.

Розроблена комплексна нейромережева модель поведінки користувачів комп'ютерних систем, що базується на нейромережевому підході і включає три компоненти: інтерактивну (прогнозну) складову, що враховує динаміку поведінки користувачів, сеансову (статистичну) складову, що враховує статистичні властивості поведінки користувача і модуль аналізу трендів, призначений для виявлення можливих змін в поведінці користувачів.

Розроблена генеративна модель поведінки користувачів, що забезпечує репрезентативні набори даних для верифікації та ідентифікації комплексної моделі.

На основі комплексної моделі поведінки користувачів практично реалізована інтелектуальна багатоагентна система моніторингу і комплексного аналізу діяльності користувачів комп'ютерних систем.

Ключові слова: НЕЙРОМЕРЕЖЕВА МОДЕЛЬ, ПОВЕДІНКА КОРИСТУВАЧІВ, ІНТЕРАКТИВНА МОДЕЛЬ, СЕАНСОВІ МОДЕЛЬ, АНАЛІЗ ТРЕНДІВ, ГЕНЕРАТИВНА МОДЕЛЬ, БАГАТОАГЕНТНА СИСТЕМА.

ABSTRACT

Diploma work «Neural-network models of behavior of computer systems users» on acquiring of educationally-qualification «Master» degree, from speciality «Computer systems and networks» with total volume 86 pages that contains 16 illustrations, 2 additions and 55 sources of information according to the list of references.

The object is to develop effective models of behavior analysis of computer systems users and their application in systems for monitoring and detection of abnormal activity.

Research methods. In the thesis work are used methods of system analysis of complex interrelated processes, neural network approach, statistical modeling, object-oriented approach to modeling and design of complex systems, agent paradigm, methods of numerical experiment.

The is developed the complex Neural network model of user behavior of computer systems based on neural network approach and includes three components: an interactive (projected) component, which takes into account the dynamic behavior of users, session (statistical) component, which takes into account the statistical properties of user behavior and trends analysis module, that is designed to detect possible changes in user behavior.

Generative model of user behavior, providing representative sets of data for verification and identification of complex models is also developed.

Based on a comprehensive model of user behavior is practically implemented intelligent multiagent system for monitoring and complex analysis of the activity of computer systems users.

Key words: NEURAL MODELS, USER BEHAVIOR, INTERACTIVE MODEL, SESSION MODEL, TREND ANALYSIS, GENERATIVE MODEL, MULTIAGENT SYSTEMS.

ЗМІСТ

ВСТУП

Актуальність роботи. Масштабне використання комп'ютерних технологій практично у всіх сферах людської діяльності приводить до все більшої уваги до самого користувача. Знання про те, які дії він виконує (або повинен виконувати), може застосовуватися в різних областях. Наприклад, в системах стеження за персоналом (Personal Security Programs), системах безпеки, при створенні призначених для користувача оточень, що персоналізуються, в Web-додатках і т. д. Тому розробка методів аналізу і моделей поведінки користувачів комп'ютерних систем є актуальною задачею.

Проте існуючі методи і підходи до аналізу поведінки користувачів комп'ютерних систем в недостатній мірі враховують всі аспекти його роботи, що обумовлене складністю людино-машинної взаємодії. Так, в деяких роботах враховуються тільки динамічні властивості поведінки користувачів, але ніяк не аналізується статистична інформація про їх діяльність. І навпаки, існують роботи, в яких модель поведінки користувачів грунтується на використанні статистичних даних, але при цьому аналіз його роботи в режимі реального часу не проводиться. При цьому в більшості робіт ніяк не враховуються можливі тренди зміни поведінки користувачів.

Отже, з огляду на те, що поведінкою користувачів комп'ютерних систем є складний процес, в даний час є актуальною розробка ефективних моделей комплексного аналізу поведінки користувачів. Вирішенню саме цієї задачі присвячена дана дипломна робота.

Мета і завдання дослідження. Метою роботи є розробка ефективних моделей аналізу поведінки користувачів комп'ютерних систем і їх застосування в системах моніторингу і виявлення аномальної діяльності.

Для досягнення поставленої мети в дипломній роботі необхідно вирішити наступні завдання:

– дослідити і класифікувати існуючі методи і підходи до побудови моделей поведінки користувачів комп'ютерних систем;

– розробити комплексну модель користувачів комп'ютерних систем, яка

враховуватиме як динамічні, такі статистичні властивості їх поведінки, а також можливі зміни в їх поведінці;

– розробити генеративну модель поведінки користувачів;

– забезпечити верифікацію розробленої моделі на реальних даних;

– розробити і реалізувати систему моніторингу діяльності користувачів комп'ютерних систем в корпоративних мережах з використанням об'єктно-орієнтованого підходу і агентної технології.

Об'єкт дослідження – процеси людино-машинної взаємодії.

Предмет дослідження – нейромережеві моделі аналізу поведінки користувачів комп'ютерних систем.

Методи досліджень. В дипломній роботі використовуються методи системного аналізу складних взаємозв'язаних процесів, нейромережевий підхід, статистичне моделювання, об'єктно-орієнтований підхід до моделювання і розробки складних систем; агентна парадигма, методи чисельного експерименту.

Наукова новизна одержаних результатів. Розроблена комплексна нейромережева модель поведінки користувачів комп'ютерних систем, що базується на нейромережевому підході і включає три компоненти: інтерактивну (прогнозну) складову, що враховує динаміку поведінки користувачів, сеансову (статистичну) складову, що враховує статистичні властивості поведінки користувача і модуль аналізу трендів, призначений для виявлення можливих змін в поведінці користувачів. На відміну від існуючих підходів, запропонована модель дозволяє забезпечити комплексний підхід до аналізу поведінки користувачів як під час його роботи (у режимі реального часу), так і після закінчення сеансу (у відкладеному режимі). Розроблена генеративна модель поведінки користувачів, що забезпечує репрезентативні набори даних для верифікації та ідентифікації комплексної моделі.

Практичне значення отриманих результатів. На основі комплексної моделі поведінки користувачів практично реалізована інтелектуальна багатоагентна система моніторингу і комплексного аналізу діяльності користувачів комп'ютерних систем. Розроблена загальна ієрархія класів для реалізації нейромережевих моделей, яка забезпечує ефективність обчислень,

масштабованість і можливість повторного використання, розроблений шаблон інтелектуального агента, що інкапсулює нейромережеві моделі.

У першому розділі роботи розглянуті області застосування моделей поведінки користувачів комп'ютерних систем, проаналізовані різні підходи до їх побудови, визначені їх переваги і недоліки. Аналіз показує, що моделі поведінки користувачів широко застосовуються для вирішення задач в різних областях. У даному розділі розглянуті існуючі методи прогнозування і класифікації, зокрема методи на основі побудови шаблонів дій користувачів, які часто повторюються, а також методи, засновані на застосуванні марківських ланцюгів і нейронних мереж. Аналіз існуючих методів прогнозування і класифікації показує, що найбільш ефективним при вирішенні складних нестаціонарних задач є підхід на основі нейронних мереж.

У другому розділі запропонована комплексна модель поведінки користувачів комп'ютерних систем, яка містить три компоненти: інтерактивну (прогнозну) складову, сеансову (статистичну) складову і модуль аналізу трендів. Інтерактивна складова грунтується на прогнозуванні дій користувача на основі попередніх, що дозволяє описати динамічні властивості його поведінки. Для прогнозування використовується нейромережевий підхід. Запропонована сеансова складова враховує статистичні параметри поведінки користувача (сигнатура користувача), які були отримані впродовж сеансу, а саме: кількість введених команд; результати інтерактивної моделі; набір комп'ютерів; тривалість сеансу; час початку сеансу. В даному випадку нейронна мережа працює як класифікатор. Також розроблена генеративна модель поведінки користувачів, що забезпечує репрезентативні набори даних для верифікації і ідентифікації комплексної моделі. Для побудови даної моделі виконувалося статистичне моделювання параметрів сеансової складової для кожного користувача: кількості команд, що вводяться, за сеанс, номери комп'ютера, тривалості і часу початку сеансу.

У третьому розділі розглянуті питання, пов'язані з програмною реалізацією комплексної моделі користувачів комп'ютерних систем. Для реалізації системи був використаний агентний підхід. Це дало можливість використовувати запропоновану систему моніторингу в гетерогенному середовищі (з різними

операційними системами і форматами даних), а також забезпечити автономність і мобільність компонентів (можливість переміщення моделі на сторону користувача). Використання агентної технології дозволило зробити систему розподіленою і легко масштабованою. В якості технології реалізації і середовища програмування мобільних агентів були вибрані, відповідно, Java і Aglets Software Development Kit (ASDK). Реалізована нейромережева система моніторингу підтвердила дієздатність і ефективність запропонованої моделі користувача.

1 АНАЛІЗ МЕТОДІВ ПОБУДОВИ І ОБЛАСТЕЙ ЗАСТОСУВАННЯ МОДЕЛЕЙ ПОВЕДІНКИ КОРИСТУВАЧІВ

1.1 Аналіз методів і підходів до аналізу поведінки користувачів

В даний час для аналізу поведінки користувачів застосовуються різноманітні методи і підходи. Проте більшою мірою всі вони засновані на аналізі і виявленні закономірностей і часто повторюваних дій користувачів з метою вирішення задач автоматизації, прогнозування його поведінки, виявлення аномалій в його роботі, автоматичній адаптації комп'ютерної системи до потреб користувачів і т. д. Дані, які використовуються для побудови моделей, враховують в основному знання про індивідуальні характеристики користувача, які визначають його поведінку. Для їх обробки можуть використовуватися як прості методи, засновані на аналізі часто повторюваних дій (history-matching methods), так і складні методи машинного навчання.

Оскільки при побудові моделей поведінки користувачів найчастіше виникають складні задачі прогнозування і класифікації, розглянемо різні підходи, які застосовуються для їх вирішення.

1.1.1 Методи прогнозування

По оцінках вітчизняних і зарубіжних дослідників число різних методів, прийомів і методик прогнозування перевищило 150 [1]. Проте число базових методів, що повторюються в різних варіаціях в інших методах, не перевищує десятка. Під методом прогнозування розуміється сукупність прийомів прогнозування, що дозволяють на основі ретроспективних даних, відомих зовнішніх і внутрішніх зв'язків підсистем, а також їх вимірювань, вивести думки певної достовірності відносно майбутнього системи [2]. Розрізняють наступні основні методи прогнозування: евристичні методи прогнозування, математичні методи тимчасової екстраполяції, математичні методи просторової екстраполяції, методи моделювання процесів розвитку, логічні і структурні методи штучного інтелекту.

Евристичні методи прогнозування.

Евристичне прогнозування полягає в інтуїтивному виборі з незліченної множини обставин найважливіших і найвирішальніших. Велика частина цієї інтуїції полягає в напівсвідомому порівнянні експертом всіх величин і варіантів, за допомогою яких швидко усувається все маловажне і неістотне [3]. Проте при цьому евристичні методи суб'єктивні і придатні тільки тоді, коли існують експерти, які знайомі з прогнозованою ситуацією.

Методи тимчасової екстраполяції.

Залежно від використовуваного математичного апарату і цільової спрямованості, математичні методи тимчасової екстраполяції можна умовно розділити на три групи: методи аналітичного прогнозування; методи імовірнісного прогнозування; методи статистичної класифікації. До методів аналітичного прогнозування багатовимірних процесів належить градієнтний метод, в рамках якого функція стану екстраполюється у напрямі вектора градієнта функції стану [4]. Існує ряд методів аналітичного прогнозування, що враховують похідні змін функції стану [5]. До таких методів відносять операторний метод, метод підсумовування похідних і т. д. [4]. До загальних недоліків методів аналітичного прогнозування слід віднести великий об'єм обчислювальних процедур при визначенні прогнозних значень параметрів, а також неточність результатів прогнозування при неправильно вибраній моделі.

Необхідність імовірнісного прогнозування багатовимірних процесів визначається сильним впливом зовнішніх і внутрішніх чинників, що мають випадковий характер [6]. Переважання випадкової складової при вимірюваннях приводить до великих випадкових змін функцій стану. Для отримання безперервного прогнозу використовуються оптимальні фільтри: фільтр Вінера-Хопфа для прогнозування стаціонарних процесів і фільтр Кальмана для нестаціонарних процесів [4]. До загальних недоліків більшості імовірнісних методів прогнозування багатовимірних процесів можна віднести: необхідність наявності представницького об'єму статистичних даних про процеси зміни параметрів; неможливість обліку стрибків на ділянці прогнозування; неможливість обійтися без математичного опису процесів зміни параметрів.

У теорії статистичної класифікації (розпізнавання образів) процес встановлення екстраполяційних зв'язків здійснюється на основі апріорної інформації [7]. Недоліком цього підходу є обов'язкова наявність апріорної інформації. По суті, необхідна вибірка даних по об'єкту одного типу з об'єктом, стан якої необхідно прогнозувати.

Прогнозування параметрів стану у вигляді тимчасової екстраполяції характеристик використовує в якості аргументу один параметр – час. Просторова екстраполяція пов'язана з прогнозуванням в просторі характеристик, і полягає в оцінці значень векторного поля за окремими спостереженнями [8]. Для вирішення цього завдання використовують різні методи апроксимації і метод багатовимірної лінійної екстраполяції (у разі наявності невеликої кількості вимірювань) [8].

Методи математичних моделей розвитку.

Як правило, виділяють три методи моделювання: фізичне, математичне і імітаційне [3]. Фізичне моделювання дозволяє відтворювати функціонування тільки окремих елементів і підсистем об'єкту із збереженням його фізичної природи. Математичне моделювання припускає опис за допомогою сукупності співвідношень – рівнянь, нерівностей, логічних умов прогнозних характеристик стану. Недоліки математичного моделювання типові для недоліків методів тимчасової екстраполяції. Через складність об'єкту математична модель функціонування може бути дуже громіздкою для її оперативного використання. Імітаційне моделювання застосовується у тому випадку, коли досліджувані процеси складні і багатообразні настільки, що математична модель стає дуже грубим наближенням до дійсності [3].

Логічні і структурні методи прогнозування.

Логічні і структурні методи прогнозування розглядаються переважно в рамках однойменних методів розпізнавання образів, де під образами маються на увазі прогнозовані явища і процеси. Виняток становить ряд методів прогнозування, найбільш поширений з яких заснований на морфологічному аналізі. Морфологічний аналіз пов'язаний з аналізом структурних взаємозв'язків між об'єктами, явищами і концепціями [1]. Логічні методи прогнозування

використовують логічні методи розпізнавання прогнозованих явищ і засновані на дискретному аналізі і численні висловів [9]. Структурні (лінгвістичні) методи прогнозування використовують спеціальні граматики, що породжують мови, які складаються з пропозицій, кожне з яких описує те, що належить до конкретного класу прогнозних станів [9]. Недоліки цих методів характерні для методів штучного інтелекту: труднощі відпрацювання мови опису прогнозованих класів, витягування інформації з бази даних, а також складність прогнозної моделі.

Таким чином, за результатами аналізу можна переконатися в перевазі використання підходу тимчасової екстраполяції при прогнозуванні складних процесів або параметрів стану великих систем. Основні недоліки даного підходу: порівняно низька оперативність обробки інформації на ЕОМ послідовного типу, неможливість вирішення слабо формалізованих завдань процедури прогнозування складних об'єктів і відносно низька точність результатів. Для компенсації перерахованих недоліків зазвичай застосовуються нейромережеві алгоритми екстраполяції в просторі ознак станів складних об'єктів, а також комбіновані алгоритми штучного інтелекту.

1.1.2 Методи класифікації

Класифікатор *к*-найближчих сусідів. В основі даного алгоритму лежить правило найближчого сусіда, що є одним з простих методів навчання на основі пам'яті [10]. Згідно цьому правилу весь минулий досвід накопичується у великому сховищі правильно класифікованих прикладів вигляду вхід-вихід: $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, де x_{i} – вхідний вектор, а d_i – відповідний йому бажаний вихідний сигнал. Тоді за наявності тестового прикладу в його околицю включається найближчий до нього приклад. Наприклад, вектор $\mathbf{x}'_N \in \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ вважається найближчим сусідом вектора xtest, якщо виконується умова $\min_i \; d(\mathbf{x}_i, \mathbf{x}_{\text{test}}) = d(\mathbf{x}'_N, \mathbf{x}_{\text{test}})$, де $d(x_{\text{i}}, x_{\text{test}})$ – Евклідова відстань між векторами x_{i} і x_{test}. Клас, до якого відноситься найближчий сусід, вважається також класом тестованого вектора x_{test}. Це правило не залежить від розподілу, який використовується при генеруванні прикладів навчання.

Проводилося формальне дослідження правила найближчого сусіда, що

використовується для вирішення задачі класифікації образів [11]. При цьому аналіз грунтується на двох наступних припущеннях.

Приклади (x_i, d_i), що класифікуються, незалежні і рівномірно розподілені відповідно до сумісного розподілу прикладу (x, d).

Розмірність навчальної множини N нескінченно велика.

Показано, що при цих двох припущеннях ймовірність помилки класифікації при використанні правила найближчого сусіда вдвічі перевищує Байєсовську ймовірність помилки. Байєсовська ймовірність помилки – це мінімальна ймовірність помилки на множині всіх правил ухвалення рішення. У цьому контексті можна вважати, що половина класифікаційної інформації для навчальної множини нескінченного розміру міститься в даних про найближчого сусіда. Це досить несподіваний результат.

Варіацією класифікатора на основі найближчого сусіда є класифікатор k-найближайших сусідів. Він описується таким чином.

а) знаходимо k класифікованих сусідів, найближчих до вектора x_{test}, де k – деяке ціле число.

б) вектор x_{test} відносимо до того класу (гіпотези), який частіше за інших зустрічається серед k-найближайших сусідів тестованого вектора.

Таким чином, класифікатор на основі k-найближайших сусідів працює подібно до пристрою усереднювання. До його недоліку можна віднести наступне: він може не врахувати одиничний “викид”. (Викид – це спостереження, яке відрізняється від номінальної моделі.)

Метод потенційних функцій базується на гіпотезі про характер функцій, що розділяють множини, які відповідають різним класам. Великою перевагою методу є те, що до нього можна звести багато відомих алгоритмів розпізнавання. У разі використання аналізу дискримінанта розділяючі поверхні визначаються скалярними функціями: $g_1(k)$, ..., $g_m(k)$, , де $k=\{k_1, ..., k_n\}$ – вектор стану системи [12]. Функції $g(k)$, названі дискримінантами, вибираються так, щоб для всіх $k \in R_\lambda$ (де R клас) виконувалася умова $g_\lambda(k) > g_r(k)$, $\lambda, r=1, ..., u, \lambda \neq r$. За допомогою функцій дискримінантів можна отримати стандартний і зручний метод завдання розділяючих поверхонь. Проте для точного визначення функцій необхідно мати

повну апріорну інформацію про об'єкти, що підлягають класифікації, що часто буває неможливо.

1.1.3 Методи на основі аналізу дій, які часто повторюються

Даний підхід заснований на побудові шаблонів часто повторюваних дій, які користувачі виконують, наприклад, в операційній системі, Web-браузері або інших програмних продуктах. Під час роботи користувача його поточна поведінка порівнюється з побудованою моделлю шаблонів. При цьому результати порівняння можуть використовуватися для автоматичного виконання набору дій, а також для прогнозування дій користувача. Основною перевагою даного підходу до побудови моделей користувачів є його простота. До його недоліків слід віднести наступні: складність побудови моделей, що персоналізуються, для кожного користувача, в ньому ніяк не враховуються взаємозв'язки між побудованими шаблонами.

Приведемо приклади робіт, які використовують даний підхід для побудови моделей користувачів. У роботах [13, 14] він застосовується для створення адаптивного помічника в операційної системи Unix. У роботах [15, 16] він використовується для вибору найбільш ймовірних URL-адресів в Web-браузерах, причому в [16] автори пропонують до того ж враховувати їх пріоритет. У роботі [17] цей підхід використовується для створення прогнозуючого призначеного для користувача інтерфейсу (predictive user interface).

1.1.4 Методи аналізу на основі марківських ланцюгів

Багато прогнозних моделей користувачів (тобто ті, що описують послідовності дій, що виконуються користувачем, з метою прогнозу) засновано на застосуванні марківських ланцюгів. Це пов'язано з наступним припущенням про характер поведінки користувачів: ймовірність виконання наступної дії залежить від декількох попередніх. Про такі процеси говорять, що вони мають "коротку" пам'ять. Це наштовхує на думку моделювати такі послідовності даних за допомогою марківських ланцюгів і прихованих марківських моделей (HMM – Hidden Markov Model). Проте, не дивлячись на те, що ці статистичні моделі здатні

описувати широкий клас послідовностей і існують ефективні процедури їх настройки, вони мають серйозні недоліки. Справа в тому, що розмір марківських ланцюгів експоненціально збільшується із зростанням їх порядку, тобто кількість станів відповідного автомата поводиться як $O(|\Sigma|^L)$, де $|\Sigma|$ – розмір алфавіту символів, а L – порядок ланцюга. Отже, практичну цінність мають тільки моделі з дуже невеликим порядком. Проте в цьому випадку вони можуть погано апроксимувати послідовності дій, що виконуються користувачами. Що стосується прихованих марківських моделей, то окрім теоретично доведеної складності їх навчання, спроби їх практичної реалізації показали необхідність вельми великих витрат на їх настройку.

Приведемо приклади робіт, які використовують даний підхід для побудови моделей користувачів.

У роботі [18] розглядається побудова змішаної (глобальної) моделі Web-користувачів і створення згодом персоніфікованої імовірнісної моделі для кожного з них. Мета даного підходу прогнозування дій користувача на основі попередніх. Для цього використовуються метод максимальної ентропії і марківські ланцюги. Ідея запропонованого методу полягає в наступному: спочатку будується глобальна модель поведінки на основі аналізу дій тих користувачів, для яких є дані (деяка апріорна інформація). Причому кожному з них присвоюється індивідуальних ваговий коефіцієнт в змішаній моделі. Потім будь-який інший користувач, що відвідав Web-вузол, відноситься до тієї або іншої групи користувачів. Тобто поведінка нового користувача або користувача з незначним набором даних (у статистичному сенсі) може бути наближена (з деякою ймовірністю) до вже відомої і, таким чином бути спрогнозованою.

У роботі [19] пропонується модель, яка використовується для прогнозування команд, що виконуються користувачем в операційній системі UNIX, а також допомагає користувачеві донабирати команди: тобто на основі k введених символів вибирати найбільш ймовірну команду. Для вирішення поставленого завдання авторами розроблений спеціальний метод покрокового імовірнісного моделювання дій, який заснований на використанні марківських ланцюгів. У даній роботі за основу взята проста марківська модель першого

порядку, тобто вона не враховує контекст довжини більший, ніж одна команда. Експерименти, які були проведені на даних 77 користувачів, показали 39,9% точність прогнозу.

У роботах [20, 21] запропонована модель користувача комп'ютерних систем, яка застосовується для виявлення аномалій в його поведінці. Для цього використовуються марківські ланцюги з порядком, що змінюється. Автори статті вказують, що застосування адаптивної настройки марківського ланцюга з пам'яттю змінної довжини дозволяє точніше підстроїтися до особливостей поведінки користувачів комп'ютерної системи і безперервно уточнювати модель, дозволяючи враховувати нестатичність поведінки суб'єктів.

1.1.5 Методи аналізу на основі нейронних мереж

У багатьох роботах для аналізу поведінки користувачів комп'ютерних систем застосовуються нейроні мережі, що забезпечує інтелектуальний і адаптивний підхід до аналізу і узагальнення даних. Ці властивості дозволяють нейронним мережам вирішувати складні завдання (у контексті побудови моделей користувачів) прогнозування, класифікації, фільтрації і т. д. Розглянемо деякі приклади використання нейромережевого підходу до створення моделей поведінки користувачів.

У роботі [22] запропонована система NNID (Neural Network Intrusion Detector), яка ідентифікує користувачів на основі обмеженої кількості команд (100 штук), введених протягом дня, і однієї нейронної мережі, яка побудована для всіх користувачів. При цьому враховується тільки кількості запусків кожної команди, а не їх послідовність. Ці кількості певним чином кодуються і є входом нейронної мережі. Якщо сеанси користувачів не потрапляють під їх нормальний шаблон, тобто користувач не ідентифікований або ідентифікований неправильно, то генерується відповідне повідомлення про атаку. Були отримані непогані результати, але для комп'ютерної системи, що складається зі всього 10 користувачів. Очевидно, що це достатньо "тепличні" умови, оскільки в реальних системах кількість користувачів може досягати декількох тисяч, причому більшість виконуватиме однотипні дії. Тому побудова однієї нейромережевої

моделі для ідентифікації всіх користувачів на основі тільки кількості команд може бути дуже скрутною.

У іншій роботі [22] ідентифікація користувача проводилася на основі послідовності команд. Нейронна мережа використовувалася для прогнозування введених команд. При цьому команди кодувалися таким чином: спочатку вони були пронумеровані в довільному порядку, а потім кожній команді ставився у відповідність вектор – двійковий запис його номера. Проте реальний вихід нейронної мережі не був двійковим: компоненти вихідного вектора складали реальні числа. Тому в даному випадку критерій аномальності був не наочним. Також варто відзначити, що модель була побудована тільки для роботи з аудит-файлами UNIX-систем.

1.2 Області застосування методів аналізу поведінки користувачів

1.2.1 Системи безпеки

Відомо, що достатньо велика кількість атак на комп'ютерну систему здійснюється внутрішніми користувачами. Деякі джерела вказують на те, що близько 80-90% атак ініціюється саме користувачами всередині комп'ютерної системи [23]. Це може відбуватися внаслідок здійснення підміни користувачів, перевищення користувачами своїх прав, використання недоліків в підсистемі захисту або ж внаслідок використання помилок в програмному забезпеченні. На жаль, традиційні методи захисту, такі як аутентифікація, обмеження прав доступу і т. д. виявляються неефективними при виявленні аномалій в роботі користувачів. Тому одним з сучасних методів вирішення цієї проблеми є використання систем виявлення і запобігання вторгненням (відповідно, IDS Intrusion detection systems і IPS Intrusion prevention systems) [24].

У загальному випадку системи IDS і IPS поділяються на такі, що використовують моделі атак [25, 26], і такі, які засновані на аналізі поведінки (системи виявлення і запобігання аномаліям) [20, 21].

При використанні методів виявлення вторгнень, заснованих на сигнатурах атак, застосовуються знання, раніше накопичені про атаки і вразливості (слабкі

місця) комп'ютерної системи. Система виявлення вторгнення містить інформацію про ці вразливості і виявляє спроби їх використання. Якщо така спроба виявлена, система генерує повідомлення про тривогу. Іншими словами, будь-яка дія, яка явним чином не визначено як атака, розглядається прийнятною. Перевагами такого підходу є теоретично низький рівень помилкових тривог. Недоліками складність збору необхідної інформації про відомі атаки, необхідність постійного оновлення баз даних вразливостей і сигнатур, практична неможливість виявлення атак, що ініціюються користувачами всередині комп'ютерних систем.

У свою чергу, методи виявлення і запобігання вторгненням на основі поведінки припускають, що вторгнення може бути виявлене на основі відхилення від нормальної або очікуваної поведінки системи або користувачів. Модель нормальної поведінки будується на основі інформації, зібраної впродовж деякого часу функціонування системи або роботи користувачів. Система виявлення вторгнень порівнює цю модель з поточною інформацією. Якщо відхилення виявлене, здійснюється сповіщення про тривогу. У системах запобігання вторгненням, до того ж, можуть здійснюватися автоматичні дії із запобігання атакам. Іншими словами, все, що не відповідає заздалегідь вивченій поведінці, розглядається як вторгнення.

Переваги методу виявлення вторгнень на основі аналізу поведінки полягають в тому, що вони можуть виявити спроби використання нової або непередбаченої вразливості, а також виявити аномальну роботу внутрішніх користувачів комп'ютерної системи. Високий рівень помилкових тривог вважається головним недоліком цих методів. Це пов'язано з тим, що неможливо закласти впродовж фази навчання всі можливі варіанти поведінки комп'ютерної системи або його користувача. Крім того, поведінка може мінятися з часом, яке приводить до потреби періодичного перенавчання профілю поведінки.

При побудові систем виявлення аномалій використовуються наступні підходи: статистичні моделі [27], виявлення аномалій з використанням експертних оцінок [28], моделі на основі марківських ланцюгів [20, 21], нейромережеві моделі [29, 30].

Так, в роботі [29] виявлення аномалій виконується на основі однієї моделі,

яка будується для всіх користувачів комп'ютерної системи. В якості профілю користувача використовується інформація про кількість введених ним команд за сеанс. При цьому для всіх користувачів використовується один і той же набір з 100 команд. Для ідентифікації користувачів застосовується апарат нейронних мереж. Так, якщо сеанс користувача не потрапляє під нормальний шаблон, тобто користувач не ідентифікований або ідентифікований неправильно, то генерується відповідне повідомлення про атаку. Серед недоліків пропонованого підходу можна виділити наступні: необхідність перенавчання моделі при додаванні нових користувачів; використання обмеженого набору команд може привести до того, що профілі користувачів відрізнятимуться трохи (особливо, якщо кількість користувачів комп'ютерної системи досягатиме декількох сотень).

У свою чергу, в роботі [30] ідентифікація користувачів здійснюється на основі послідовності команд. Для кожного користувача будується нейронна мережа, яка навчається прогнозувати наступну команду даного користувача на основі декількох попередніх. Якщо команди, отримані в результаті нейромережевого прогнозу, істотно відрізняються від реально введених користувачем, то можна зробити висновок про наявність аномалії в його поведінці.

1.2.2 Системи стеження за персоналом

Сьогодні є популярними так звані системи стеження за персоналом (Personal Security Programs), які в основному використовуються комерційними компаніями для спостереження за роботою своїх співробітників. Застосування таких систем дозволяє виявити зловмисників при просочуванні інформації, або ж визначити, чи використовують співробітники робочі станції в своїх особистих цілях. Такі програми, як PC Spy [31], Inlook Express [32], Paparazzi [33] дозволяють “захоплювати” і зберігати зображення (копії екранів), на яких можна бачити, які програми запускаються користувачами: броузери із завантаженими Web-сторінками, вікна з електронними повідомленнями і т. д. Тобто в даному випадку в якості профілю користувача використовується графічна інформація про запущені віконні додатки. Проте такі системи володіють рядом недоліків, серед

яких слід виділити великий об'єм даних, що зберігаються, і необхідність ручної настройки частоти “захоплення” копій екрану. Тобто, якщо частота збереження даних невелика, то виявити зловмисника буде складно. Інакше, в системі доведеться зберігати велику кількість зображень (копій екрану), що приведе до збільшення навантаження на комп'ютерну мережу.

1.2.3 Створення оболонки для користувача

Моделі поведінки користувачів широко застосовуються при створенні так званих призначених для користувача оболонок, що персоналізуються. Дані оболонки використовуються з метою автоматичної адаптації середовища до поведінки користувача, що дозволяє підвищити ефективність його роботи з різними програмами.

Так, в роботах [19, 34] пропонується адаптивний помічник командного рядка (command-line assistant), який прогнозує команди, що виконуються користувачем в операційній системі Unix, а також дозволяє йому донабирати команди. Для цього використовується спеціально розроблений алгоритм IPAM (Incremental Probabilistic Action Modeling – покрокове імовірнісне моделювання дій), точність прогнозу якого досягає 39,9%. До недоліку цього методу можна віднести те, що в ньому ніяк не враховуються опції команд, що є істотним моментом в роботі користувачів під управлінням даної операційної системи.

У роботах [35, 36] пропонуються спеціально розроблені методи машинного навчання (machine learning techniques) для здійснення прогнозу команд користувачів. Отримані достатньо непогані результати (точність прогнозу в деяких випадках досягає 70-80%) хоча і на невеликому об'ємі даних реальних користувачів. Проте така висока точність досягнута за рахунок використання потужної надбудови над операційною системою, яка дозволяє збирати різноманітну інформацію про ввід/вивід (обмін) даних, який здійснюється запущеними командами (наприклад, читання файлу).

У роботах [13, 37] використовуються простіші моделі, в яких профіль користувача враховує тільки часто повторювані набори дій. Наприклад, в [13] даний підхід застосовується з метою створення адаптивного помічника, який

виводить перелік найбільш ймовірних дій, які будуть виконані користувачем в командному рядку операційної системи Unix.

1.2.4 Web-навігація

Моделі поведінки користувачів широко застосовуються в Web-додатках, зокрема для створення адаптивних Web-вузлів [15, 16, 38].

Так, в роботі [15] методологія, заснована на аналізі часто повторюваних дій, використовується для вибору URL-адресів в Web-браузерах. У свою чергу, в роботі [16] пропонується покращений метод, який враховує до того ж пріоритет URL-адресів. Пріоритет обчислюється як зважена сума наступних характеристик: кількості переглядів даної Web-сторінки, відносної частоти її переглядів і кількості її переглядів в послідовностях. До недоліку запропонованого підходу можна віднести те, що в ньому ніяк не враховуються взаємозв'язки між Web-сторінками, які відвідує користувач. Тобто ніяк не враховується контекст.

У роботі [38] пропонується використовувати спеціальний додаток WebWatcher, який супроводжує користувача від сторінки до сторінки під час перегляду Web-ресурса і прогнозує посилання, які він вибере. Для цього використовуються знання про область інтересів конкретного користувача, а також докладна інформація про вміст Web-сторінок і взаємозв'язки між ними. Результати експериментів, які були проведені показали точність 42,2%. Варто при цьому відзначити, що запропонований підхід враховує тільки поточні інтереси користувача і ніяк не враховує історію його поведінки.

У роботі [39] пропонується підхід, що дозволяє стежити за діями користувача під час перегляду Web-стрінок. На основі отриманої інформації про його роботу будується відповідний граф (який фактично є моделлю користувача). Для витягування корисної інформації з графа використовується спеціально розроблена мова MINT (подібна SQL).

У роботі [40] пропонується розробка адаптивного Web-вузла, який формує «на льоту» Web-сторінки релевантні інтересам відвідувачів. Для цього використовується концептуальний метод витягування знань, що аналізує аудит-файли Web-сервера.

У роботі [18] розглядається побудова змішаної (глобальною) моделі (mixture model) Web-користувачів і створення згодом персоніфікованої імовірнісної моделі для кожного з них. Мета даного підходу прогнозування дій користувача (таких як перегляд документів або пошук) на основі попередніх. До недоліків даного підходу можна віднести використання ресурсоємних статистичних методів, необхідність перенавчання моделі при додаванні нових користувачів, а також практична неможливість застосування в режимі реального часу. При цьому для збору даних про роботу користувачів автори використовують механізм cookies (збереження даних в браузері на стороні клієнта), який в загальному випадку може бути відключений користувачем. Слід також відзначити якусь евристику при визначенні “сеансу” користувача якщо відвідувач Web-вузла не виконує дії впродовж 300 секунд, вважається початок нового сеансу (причому не описано, з яких міркувань визначається це число).

Моделі користувачів також застосовуються для розробки систем адаптивного інформаційного пошуку, зокрема в мережі Internet [41, 42]. Наприклад, в роботі [41] пошук інформації здійснюється на основі формальної контекстної моделі користувача. Дана модель є багатошаровою семантичною мережею, кожен шар якої відповідає деякому контексту, а вузли ключовим словам. Запропонована система реалізована за допомогою агентного підходу. У роботі [42] описується мультиагентна система пошуку на основі моделей користувачів. Запропонована система містить два типи агентів, що взаємодіють між собою: для фільтрації інформації (цей агент відповідальний за побудову і підтримку профілю користувача) і виявлення нових властивостей в поведінці користувачів (адаптація профілю користувача до різних інформаційних ресурсів). При цьому агенти «еволюціонують» шляхом застосування генетичних алгоритмів (клонування, кросовер, мутації).

1.2.5 Системи дистанційного навчання

Моделі поведінки користувачів активно використовуються в системах дистанційного навчання. Це обумовлено необхідністю адаптації системи і учбового процесу до потреб тих, що навчаються [43].

Наприклад, в роботі [43] з метою ефективного обміну інформацією використовується модель як учня, так і викладача. Модель учня грунтується на інформації про його взаємодію з системою у минулому (про пройдений матеріал, досконалі помилки при виконанні тих або інших завдань) і є комбінацією стереотипних (stereotype) моделей, що перекриваються (overlay). Стереотипна модель застосовується з метою класифікації знань і навиків студента на різні рівні: новачок, початківець, експерт. У свою чергу, модель, що перекривається, є набором пар «поняття/значення» наочної області. Модель учня використовується для виявлення тенденцій в здійсненні однотипних помилок, для формування звіту про успішність кожного учня, для перевірки, наскільки учень засвоює матеріал і т. д. Модель викладача використовується для забезпечення допомоги викладачеві при організації навчального процесу.

У роботі [44] розглядається завдання прогнозування поведінки студентів в навчальних системах. Для цього використовується дуальна модель (dual model), що складається з двох частин: ранньої моделі (fresh model) і розширеної моделі (extended model). Для побудови ранньої моделі використовуються найбільш «свіжі» дані про роботу студента. У свою чергу, розширена модель враховує всю історію поведінки користувача в системі навчання. Прогнозування дій студента здійснюється на основі обох моделей.

1.2.6 Виявлення шахрайства з кредитними картками

В даному випадку аналіз поведінки користувачів використовується з метою виявлення аномалій, пов'язаних з використанням банківських кредитних карток. При цьому при вирішенні цієї задачі необхідно враховувати наступні чинники: наявність обмеженого часу для ухвалення рішення і необхідність обробки великої кількості інформації [45]. Так, в роботі [46] для виявлення шахрайського використання кредитних карток застосовуються експертні системи. У роботах [46, 48] запропонований підхід, що базується на використанні нейронних мереж на основі радіально-базисних функцій. У статті [45] використаний нейромережевий класифікатор, що дозволяє відносити поточну «поведінку» кредитної картки до одного з класів (нормальна, аномальна). Нейромережевий підхід також

застосовується і в публікації [49] для побудови системи витягування даних. У роботі [50] розроблений спеціальний підхід метанавчання, що дозволяє об'єднувати результати різних класифікаторів, кожен з яких вказує ступінь аномальності використання кредитної картки користувачем.

1.3 Постановка задачі дослідження

Аналіз існуючих моделей поведінки користувачів, проведений в попередньому розділі, показав, що в більшості випадків вони є простими і не дозволяють описати всі аспекти поведінки. Так, в багатьох моделях виявлення аномальної діяльності користувачів здійснюється в рамках одного тимчасового масштабу. Наприклад, існують моделі, які враховують тільки динамічні властивості поведінки користувачів, але при цьому не аналізують його статистичні властивості. Тобто моніторинг і виявлення аномалій відбувається в режимі реального часу під час сеансу роботи користувача. І навпаки, існують моделі, які засновані тільки на аналізі статистичної інформації про діяльність користувачів за сеанс в цілому. У таких моделях виявлення аномальної діяльності здійснюється у відкладеному режимі. У більшості робіт також не ставиться питання про виявлення змін поведінки користувачів, не пов'язаних з аномалією, і визначення моменту часу, коли модель необхідно перенастроювати (тобто адаптувати до зміни поведінки користувача).

Тому актуальною є задача розробки моделей поведінки користувачів комп'ютерних систем, які враховуватимуть як динамічні, так і статичні властивості поведінки користувачів, а також можливі тренди їх поведінки.

У дипломній роботі ставиться задача розробки комплексного підходу до аналізу поведінки користувачів з метою виявлення аномалій в його роботі. Даний підхід повинен враховувати динамічні і статистичні властивості поведінки, а також можливі зміни в поведінці, не пов'язані з аномаліями.

Під динамічними властивостями розумітимемо послідовність дій користувача (запуск процесів) в перебігу одного сеансу роботи.

Статистичними (або статичними) властивостями називатимемо інтегральні

характеристики роботи користувача в перебігу сеансу.

При побудові моделей поведінки користувачів можна виділити наступні загальні етапи:

– збір і попередня обробка даних про роботу користувачів;

– аналіз даних з метою виділення інформативних ознак або зменшення розмірності даних (створення так званого профілю користувача);

– розробка методів обробки даних і побудова моделі;

– верифікація моделі і інтерпретація отриманих результатів.

У дипломній роботі будуть розглянуті всі перераховані етапи.

Після того, як для кожного користувача комп'ютерної системи побудована модель, виявлення аномальної діяльності повинне відбуватися таким чином: якщо результати поточної роботи користувача відповідають раніше побудованій моделі, то таку поведінку можна вважати нормальною. Інакше аномальною.

У даному розділі розглянуті області застосування моделей поведінки користувачів комп'ютерних систем, проаналізовані різні підходи до їх побудови, визначені їх переваги і недоліки.

Аналіз показує, що моделі поведінки користувачів широко застосовуються для вирішення задач в різних областях, наприклад, в системах безпеки, в системах стеження за персоналом, в Web-додатках, в системах дистанційного навчання, для виявлення шахрайства, пов'язаного з використанням банківських кредитних карток і т. д. При цьому для аналізу дій користувача застосовуються різноманітні методи і підходи. Практично всі вони засновані на виявленні закономірностей в його роботі шляхом прогнозування або класифікації (наприклад, нормальна/аномальна поведінка). У даному розділі розглянуті існуючі методи прогнозування і класифікації, зокрема методи на основі побудови шаблонів дій користувачів, які часто повторюються (наприклад, в операційній системі, Web-браузері або інших програмах), а також методи, засновані на застосуванні марківських ланцюгів і нейронних мереж. Аналіз існуючих методів прогнозування і класифікації показує, що найбільш ефективним при вирішенні складних нестаціонарних задач є підхід на основі нейронних мереж.

Існуючі моделі в більшості випадків є простими і не дозволяють описати всі

аспекти поведінки користувачів комп'ютерних систем. Так, розроблені моделі, які враховують тільки динамічні властивості поведінки користувачів, але при цьому не аналізують його статистичні властивості. І навпаки, існують моделі, які засновані на аналізі статистичної інформації про діяльність користувачів, але при цьому не дозволяють аналізувати поведінку під час сеансу його роботи (тобто в режимі реального часу). У більшості підходів також не розглядається питання про можливість зміни поведінки користувачів і визначення моменту часу, коли модель необхідно перенастроювати.

Тому актуальною задачею є розробка ефективних методів системного аналізу і моделей поведінки користувачів комп'ютерних систем, які враховуватимуть як динамічні, так і статичні (інтегральні) властивості поведінки користувачів, а також можливі тренди його поведінки. У даній роботі пропонується комплексний підхід до аналізу поведінки користувачів з метою виявлення аномалій в його роботі, який дозволяє врахувати всі ці аспекти.

2 НЕЙРОМЕРЕЖЕВІ МОДЕЛІ ПОВЕДІНКИ КОРИСТУВАЧІВ КОМП'ЮТЕРНИХ СИСТЕМ

2.1 Комплексна модель поведінки користувачів комп’ютерних систем

У даному розділі обгрунтовується комплексний підхід до аналізу поведінки користувачів комп'ютерних систем з метою моніторингу і виявлення аномальної діяльності. Пропонується загальна структура комплексної моделі і опис її складових (інтерактивної, сеансової і модуля аналізу трендів), аналізуються переваги запропонованого підходу в порівнянні з існуючими моделями і підходами.

2.1.1 Структура комплексної моделі поведінки користувача

Для адекватного опису поведінки користувачів комп'ютерних систем і виявлення аномалій в його роботі пропонується використовувати комплексний підхід, що включає три компоненти:

- інтерактивну (прогнозну) складову, що описує роботу користувача під час сеансу на рівні виконуваних команд (або процесів, що запускаються);
- сеансову (статистичну) складову, яка заснована на аналізі інтегральних даних про роботу користувача за сеанс в цілому;
- модуль аналізу трендів, що дозволяє виявити можливі тренди в роботі користувача.

Запропонована структура комплексної моделі представлена на рисунку 2.1.

Інтерактивну і сеансову складові комплексної моделі пропонується будувати на основі нейронних мереж [10]. Застосування нейронних мереж забезпечує інтелектуальний і робастний підхід до аналізу і узагальнення даних, отриманих під час роботи користувача. У загальному випадку існують різні нейромережеві парадигми. Наприклад, асоціативні мережі, які використовуються для вирішення задач асоціативної пам'яті і відновлення зашумленних образів, мережі Кохонена, які використовуються для кластеризації образів, мережі на основі радіальних базисних функцій (РБФ), призначені для вирішення задач

класифікації образів і т. д.

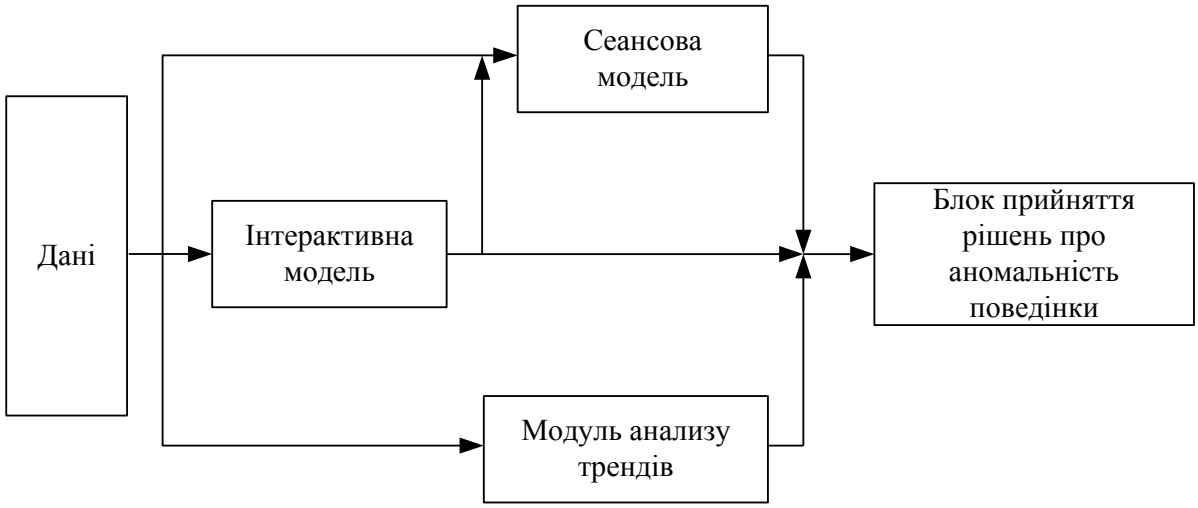


Рисунок 2.1 – Структура комплексної моделі поведінки користувача

Для вирішення задач дипломної роботи якнайкращим вибором є багатошарова мережа прямого розповсюдження інформації. Використання мереж даного виду обумовлене тим, що згідно теореми Колмогорова вони є універсальними апроксимаціями [10] і можуть ефективно застосовуватися як для вирішення задач прогнозування, так і класифікації. Застосування нейронних мереж одного і того ж виду в різних складових комплексної моделі також забезпечує переваги при реалізації моделі. Реалізувавши одну нейромережеву парадигму, її можна використовувати як в інтерактивній, так і в сеансовій складовій, що дозволяє забезпечити робастність пропонованого підходу.

Конкретний вид комплексної моделі користувача обумовлює використання різних типів нейронних мереж прямого розповсюдження. У разі інтерактивної складової нейронна мережа прогнозує команди користувача на основі попередніх. Відповідно вона використовується як інтерполятор. У разі сеансової складової нейронна мережа використовується як класифікатор і на основі інтегральних даних, які були отримані за сеанс, визначає, на скільки активність користувача відповідає раніше побудованій моделі.

2.1.2 Нейронні мережі прямого розповсюдження інформації

Штучні нейронні мережі є спрощеною моделлю нервової тканини живих організмів, клітини якої є прообразом базових компонентів сучасних

нейрокомп'ютерів [10]. Свою силу нейронні мережі черпають, по-перше, з розпаралелювання обробки інформації і, по-друге, із властивості узагальнення. Під терміном узагальнення розуміється властивість отримувати обгрунтований результат даних, які не зустрічалися в процесі навчання. Ці властивості дозволяють нейронним мережам вирішувати складні задачі прогнозування, класифікації, обробки зображень, управління і т. д. При цьому використання нейронних мереж забезпечує наступні корисні властивості систем: нелінійність, відображення вхідної інформації у вихідну, адаптивність, відмовостійка, масштабованість.

Найбільш універсальним видом нейронних мереж є мережа прямого розповсюдження. Нейронна мережа прямого розповсюдження є багатошаровою структурою взаємозв'язаних простих оброблювальних елементів (виконавчих нейронів). Зв'язки між елементами одного шару і зворотні зв'язки в мережі відсутні (рисунок 2.2).

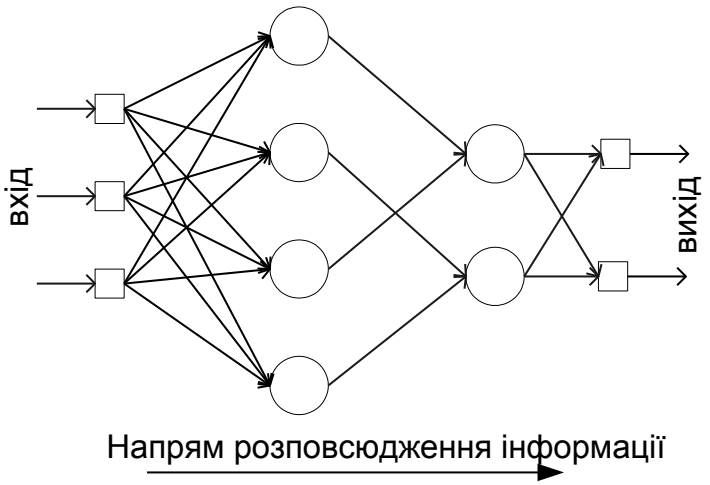


Рисунок 2.2 – Приклад архітектури штучної нейронної мережі

Для нейронних мереж прямого розповсюдження інформація від виходу шару *n* до виходу шару *n+1* розповсюджується таким чином:

$$y_j^{n+1} = f(s_j^{n+1}), y^{n+1} = f(S^{n+1}), \quad (2.1)$$

де *n*+1 – номер шару;

j – індекс нейрона в шарі $n+1(j=\overline{1,N}_{n+1})$;

N_{n+1} – число нейронів в шарі *n+1;*

f – нелінійна активаційна функція шару *n+1;*

y_j^{n+1} – вихід *j*-го нейрона шару *n+1;*

s_j^{n+1} – постсинаптичний потенціал *j*-го нейрона, який обчислюється по наступній формулі:

$$s_j^{n+1}=\sum_{k=1}^{N_n} W_{jk}^{n+1} y_k^n + b_j^{n+1},\ S^{n+1}=W^{n+1}\cdot \tilde{y}^n, \qquad (2.2)$$

де W_{jk}^{n+1} – ваговий коефіцієнт зв'язку *k*-го нейрона шару *n* з *j*-м нейроном шару *n+1*;

y_k^n – вихід *k*-го нейрона шару n;

$\tilde{y}^n$ – розширений вектор з врахуванням Bias-нейрона;

b_j^{n+1} – поріг *j*-го нейрона шару *n+1*;

W^{n+1} – матриця $d_n \times d_{n+1}$.

В якості активаційної функції використовуватимемо сигмоїдную функцію, яка має наступний вигляд: $f(x)=\frac{1}{1+e^{-\alpha x}}$ ($f(x)\in(0,1)$). Її вибір обгрунтований тим, що вона є нелінійним порогом, є безперервною і такою, що диференціюється. При цьому похідна визначається через саму функцію: $f'(x)=\alpha f(x)[1-f(x)]$. Параметр α визначає "крутизну" функції.

Для нейронної мережі можна виділити наступні режими функціонування:

- навчання – ідентифікація вільних коефіцієнтів мережі;
- тестування – перевірка роботи мережі на незалежній вибірці;
- прогін – використання мережі на невідомих даних.

Перейдемо тепер до опису складових комплексної моделі.

2.1.3 Структура інтерактивної моделі

В результаті аналізу файлів аудиту для різних користувачів виявилось, що кожен користувач комп'ютерної системи володіє характерною поведінкою, яка полягає у виконанні характерних послідовностей команд в різних сеансах. Для різних користувачів така поведінка різна, але для кожного користувача можна виявити закономірності, характерні тільки для нього. Саме на цьому і засновано використання інтерактивної моделі. Вона забезпечує прогнозування дій користувача на основі попередніх команд, тим самим описуючи динамічні властивості поведінки користувача. Для цього використовується нейромережевий підхід. Це обумовлено тим, що поведінкою користувача є складний нелінійний процес, а також необхідністю прогнозування в режимі реального часу і виявлення прихованих закономірностей в його роботі. У основу інтерактивної моделі поведінки користувачів покладена нейронна мережа прямого розповсюдження, опис якої приведений в параграфі 2.1.2.

При реалізації інтерактивної моделі для кожного користувача будується нейронна мережа, яка навчається так, щоб при подачі на вхід мережі послідовності з декількох команд на виході отримувати наступну команду. В даному випадку нейронна мережа використовується в режимі інтерполятора.

Нехай s_t (де $t \in \{1, 2, \ldots\}$ – номер сеансу) – деякий сеанс роботи користувача, для якого є наступна послідовність виконуваних ним команд:

$$\mathbf{c}^{s_t} = c_1^{s_t}, c_2^{s_t}, \ldots, c_{N_{s_t}}^{s_t},$$

де N_{s_t} – кількість команд, введених за сеанс *st*;

$c_i^{s_t} \in A$ – десятковий номер *i*-ої введеної команди сеансу st;

A – алфавіт команд.

Перш ніж подати послідовність із m команд $c_{i-1}^{s_t}, c_{i-2}^{s_t}, \ldots, c_{i-m}^{s_t}$ на вхід нейронної мережі, використовується бінарне кодування. Це зроблено з метою збільшення інформаційної ємкості вхідних образів. Для кожної команди

$c_{i-k}^{s_t}$ $k=\overline{1,m}$ її десятковий номер перетворюється двійкове число, яке складається з q біт. Тобто, кожній команді ставиться у відповідність бінарний вектор $\tilde{\mathbf{c}}_{i-k}^{s_t} \in 0,1^{q}\ (k=\overline{1,m})$. Після цього послідовність із m команд кодується шляхом об'єднання (конкатенації) побудованих двійкових векторів $\tilde{\mathbf{c}}_{i-k}^{s_t}$ кожної команди: $\mathbf{x}_i = \tilde{\mathbf{c}}_{i-1}^{s_t}, \tilde{\mathbf{c}}_{i-2}^{s_t}, \ldots, \tilde{\mathbf{c}}_{i-m}^{s_t}$. Таким чином, результат роботи нейронної мережі при виконанні *i-1* команди визначається залежністю (рисунок 2.3):

$$\tilde{c}_i^{s_t} = F(\mathbf{x}_i), \mathbf{x}_i = \tilde{\mathbf{c}}_{i-1}^{s_t}, \tilde{\mathbf{c}}_{i-2}^{s_t}, \ldots, \tilde{\mathbf{c}}_{i-m}^{s_t} \ ,$$

де F – нелінійне перетворення, яке здійснюється нейронною мережею згідно формул (2.1) і (2.2);

x_i, $\tilde{c}_i^{s_t}$ – вхід і вихід мережі, відповідно (в даному випадку розмірність вектора x_i складає $q*m$, а $\tilde{c}_i^{s_t}$ - 1);

m – глибина пам'яті.

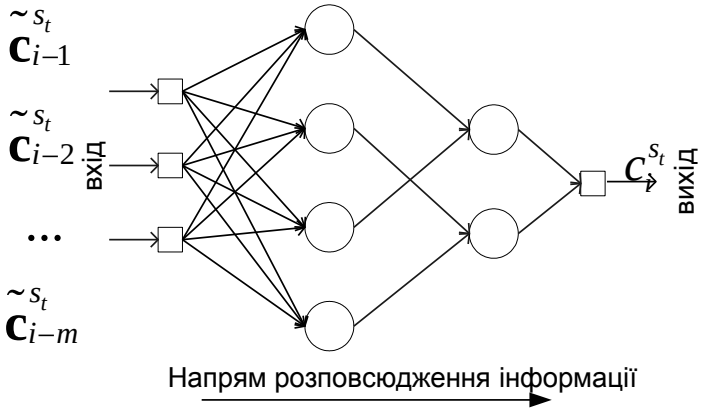


Рисунок 2.3 – Структура інтерактивної моделі

На основі кількості команд, які були правильно спрогнозовані нейронною мережею, робиться висновок про те, чи відповідає поточна поведінка користувача раніше побудованої моделі.

Нехай

$$\Theta(i) = \frac{1}{(i-m)} \sum_{j=m+1}^{i} \chi(c_i^{\sim s_t}, c_i^{s_t}) \qquad (2.3)$$

де $\chi(c_i^{\sim s_t}, c_i^{s_t}) = \begin{cases} 1, \textit{якщо } c_i^{\sim s_t} = c_i^{s_t} \\ 0, \textit{в іншому випадку} \end{cases}$

Величина $\Theta(i) \in [0; 1]$ визначає відносне число вірно спрогнозованих команд нейронною мережею до моменту введення i команд. Якщо $\Theta(i) < \Theta'$, тобто значення $\Theta(i)$ менше деякого порогу, то поведінку користувача слід вважати аномальною, інакше нормальною.

При побудові інтерактивної моделі необхідно враховувати, що користувачам властиво змінювати поведінку з часом. Тому з метою забезпечення адаптації до їх поведінки нейронну мережу слід періодично донавчати. Саме для цього і призначений модуль аналізу трендів, який буде описаний нижче.

Слід зазначити наступні переваги нейромережевої моделі користувачів в комп'ютерних системах на основі послідовності команд:

– незалежність від кількості користувачів в системі, оскільки кожен користувач розглядається окремо;

– можливість виявлення прихованих закономірностей в поведінці користувача завдяки використанню інформації не тільки про те, які команди були введені, але і про послідовності цих команд;

– адаптація до зміни поведінки користувачів.

2.1.4 Структура сеансової моделі

На відміну від інтерактивної моделі, сеансова грунтується на використанні інтегральних (статистичних) даних, отриманих під час роботи користувача за сеанс в цілому. При цьому враховується наступна інформація:

$$n_{s_t}, o_{s_t}, h_{s_t}, d_{s_t}, s_{s_t} \ , \qquad (2.4)$$

де n_{s_t} – кількість команд, які були виконані користувачем протягом сеансу;

$o_{s_t} = \Theta(N_{s_t})$ – результати інтерактивної моделі, тобто відносна кількість правильно спрогнозованих команд за сеанс (де $\Theta(N_{s_t})$ визначається співвідношенням (2.3));

h_{s_t} – номер комп'ютера в мережі, за яким працював користувач;

d_{s_t} – тривалість сеансу;

s_{s_t} – час початку сеансу.

Цей набір даних використовується в якості вхідних ознак для виявлення нормальної або аномальної роботи користувача за сеанс. Як і у разі інтерактивної моделі, для вирішення задачі класифікації використовується нейронна мережа прямого розповсюдження. Тобто для кожного користувача комп'ютерної системи будується нейронна мережа, яка навчається так, щоб на основі статистичної інформації за сеанс відносити поведінку користувача до класу нормального або аномального. При цьому очікуваний вихід нейронної мережі може приймати два значення: 1 для нормальної поведінки користувача і 0 для аномального, тобто нейронна мережа працює як класифікатор.

Нехай, як і раніше, s_t ($t \in \{1, 2, \ldots\}$) – деякий сеанс роботи користувача, після закінчення якого є наступний набір даних: $n_{s_t}, o_{s_t}, h_{s_t}, d_{s_t}, s_{s_t}$. Тоді вихід нейронної мережі після закінчення сеансу s_t визначається наступним співвідношенням (рисунок 2.4):

$$\Delta_{s_t} = F(x_{s_t}), \quad \mathbf{x}_{s_t} = (n_{s_t}, o_{s_t}, h_{s_t}, d_{s_t}, s_{s_t}),$$

де F – нелінійне перетворення, яке здійснюється нейронною мережею згідно формул (2.1) і (2.2);

$\mathbf{x}_{s_t}$ Δ_{s_t} – вхід і вихід мережі, відповідно (в даному випадку розмірність вектора $\mathbf{x}_{s_t}$ складає 5, а Δ_{s_t} – 1);

параметри n_{s_t}, o_{s_t}, h_{s_t}, d_{s_t} s_{s_t} визначені в співвідношенні (2.4).

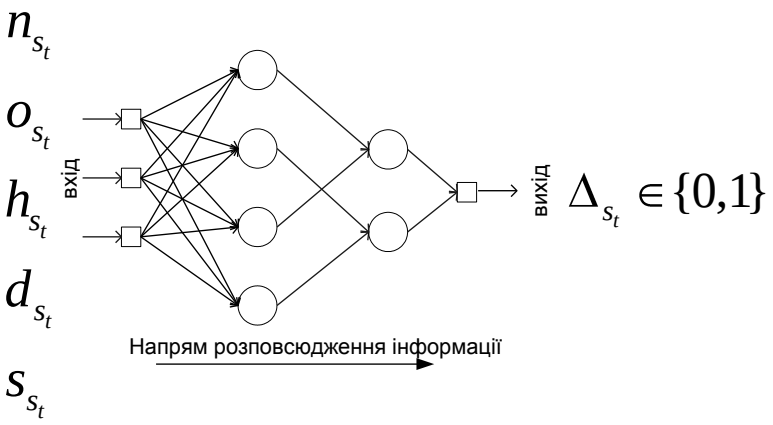


Рисунок 2.4 – Структура сеансової моделі

При реалізації сеансової моделі застосовувалося наступне кодування вхідної інформації:

– для кількості команд використовувалося десяткове кодування;

– результати інтерактивної моделі представлялися числом в проміжку [0, 1], що визначає процентне співвідношення правильно прогнозованих команд;

– набір хостів кодувався цілими числами 1, 2, 3 ...;

– тривалість сеансу користувача вимірювалася в хвилинах і нормувалася на 8 годин (тривалість робочого дня);

– час початку сеансу представлявся цілими числами від 0 до 23 і нормувалося на 24.

У роботі [10] показано, що якщо при навчанні нейронної мережі бажаний вихід приймає два значення (наприклад, 0 і 1; тобто нейронна мережа розділяє вхідний простір на два класи), то при подачі на її вхід незалежного образу на виході буде отримана ймовірність приналежності цього образу до одного або іншого класу. Таким чином, значення Δ_{s_t} належатиме відрізку [0; 1] і визначатиме ймовірність нормальної (відповідної моделі) поведінки користувача.

Слід виділити наступні переваги сеансової моделі користувача:

– незалежність від кількості користувачів в системі, оскільки для кожного користувача побудована своя нейросетевая модель;

– сеансова модель користувача враховує статистичні параметри, що дозволяє виявляти аномалії, не виявлені інтерактивною моделлю користувача;

– адаптація до зміни поведінки користувачів.

2.1.5 Модуль аналізу трендів

З практики ясно, що поведінка користувача не є стаціонарним процесом і з часом може змінюватися (зазвичай впродовж 2-3 місяців). Це може відбуватися з різних причин, наприклад, внаслідок використання нових версій програмного забезпечення. В цьому випадку, навчені на застарілих даних інтерактивна і сеансова моделі адекватно не описуватимуть поведінку користувача, і поведінка, що змінилася, інтерпретуватиметься як аномальна. Відповідно, ефективність використання цих моделей знизиться. Тому в комплексну модель поведінки користувача пропонується додати модуль аналізу трендів, що забезпечує виявлення «трендів» поведінки і що надає додаткове підтвердження аномальності поведінки користувачів. Проте при цьому необхідно враховувати, що аномальну поведінку також можна інтерпретувати як зміна поведінки. Тому при побудові модуля аналізу трендів виходитимемо з наступних припущень:

– вважатимемо, що аномальна поведінка користувача характеризується різкими змінами і швидкоплинністю.

– у свою чергу, природна зміна поведінки відбувається впродовж декількох сеансів і не характеризується різкими перепадами.

Нехай A – алфавіт команд деякого користувача (тобто набір всіх команд, які виконувалися впродовж сеансів $s_t\ _{t=1}^{T}$). Передбачається, що за цей проміжок часу поведінка користувача не змінювалася. Позначимо за допомогою N – кількість команд в алфавіті. При цьому кожній команді поставимо у відповідність номер від 1 до N. Номер N+1 алфавіту зарезервуємо за всіма тими командами, які не були виконані під час сеансів $s_t\ _{t=1}^{T}$. Нехай s_t ($t>T$) – поточний сеанс роботи користувача, для якого є наступна послідовність виконаних ним команд:

$$\mathbf{c}^{s_t} = c_1^{s_t}, c_2^{s_t}, \ldots, c_{N_{s_t}}^{s_t} .$$

Тоді $c_i^{s_t} = N+1$, якщо $c_i^{s_t} \notin \{1,...,N\}$.

Для того, щоб визначити, чи змінилася поведінка користувача, після закінчення сеансу s_t будується вектор $g(s_t)$, компоненти якого визначаються таким чином:

$$g_j(s_t) = \begin{cases} 1, \text{ якщо існує таке } k = 1, N_{s_t}, \text{ що } c_k^{s_t} = j,\ j \in A \\ 0, \text{ в іншому випадку} \end{cases}. \qquad (2.5)$$

Тобто якщо певна команда була виконана під час сеансу s_t, то значення відповідної компоненти вектора $g(s_t)$ стає рівним 1, інакше 0. Приведемо приклади векторів $g(s_t)$, побудованих для різних сеансів одного і того ж користувача:

$g(s_1)$ = (1 1 0 1 1 0 1 1 1 0 1 1 1 0 0 0 0 0 0 1 0 1 …)
$g(s_2)$ = (1 0 0 1 0 0 1 1 1 0 1 1 1 0 0 0 1 0 0 1 0 1 …)
$g(s_3)$ = (1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 0 1 0 1 …)

Потім цей вектор $g(s_t)$ попарно порівнюється з аналогічними векторами, побудованими для попередніх сеансів s_{t-1}, s_{t-2}, …, s_{t-l} (у даній роботі бралося $l = 5$). Як міра порівняння використовується відстань Хеммінга:

$$\aleph(\mathbf{g}(s_{t'}),\ \mathbf{g}(s_{t''})) = \sum_{j=1}^{N} \chi(g_j(s_{t'}),\ g_j(s_{t''})), \qquad (2.6)$$

де $\chi(g_j(s_{t'}), g_j(s_{t''})) + \begin{cases} 1, \text{ якщо } g_j(s_{t'}) \neq g_j(s_{t''}) \\ 0, \text{ в іншому випадку} \end{cases}$.

Тобто величина $\aleph$ визначає число компонентів двох векторів, значення яких різні. Отримавши в результаті порівняння l значень, обчислюємо середнє і нормуємо його на загальну кількість команд в алфавіті:

$$H_t = \frac{1}{N}\left(\frac{1}{l}\sum_{k=1}^{l}\aleph(\mathbf{g}(s_t),\ \mathbf{g}(s_{t-k}))\right). \qquad (2.7)$$

Якщо поведінка користувача не змінилася і не аномальна, тоді вектор $g(s_t)$ трохи відрізнятиметься від попередніх. Відповідно, значення параметра H_t буде невеликим (меншим деякого порогу H_*: $H_t < H_*$). І навпаки, якщо спостерігається аномальна поведінка користувача, вектор $g(s_t)$ значно відрізнятиметься від всіх останніх і значення параметра H_t буде великим (більшим деякого порогу H_*: $H_t < H_*$). Якщо ж $H_t \in (H_*; H^*)$, можна говорити про природну зміну поведінки користувача.

Таким чином, критерій зміни поведінки користувача можна сформулювати в наступному вигляді: для даного (поточного) сеансу користувача s_t на основі формул (2.5)–(2.7) обчислюється значення параметра H_t. Якщо $H_t < H_*$, вважається, що поведінка не змінилася, якщо $H_* < H_t < H^*$ – змінилася, якщо ж $H_t > H^*$ – тоді поведінка аномальна.

2.2 Комплексна модель користувача як асоціативна машина

Виходячи з приведеного опису модулів комплексної моделі, її загальна структура (див. рисунок 2.1) може бути конкретизована (рисунок 2.5).

Таку структуру прийнято називати асоціативною машиною (committee machine) [10]. Асоціативні машини зазвичай застосовуються для вирішення складних задач шляхом їх розбиття на множину невеликих і простих задач з подальшим об'єднанням отриманих рішень. При навчанні обчислювальна простота досягається за рахунок розподілу задачі навчання серед множини так званих експертів. Асоціативна машина інтегрує знання, накопичені експертами, в загальне рішення, яке має пріоритет над кожним рішенням окремого експерта.

Залежно від підходів для об'єднання рішень окремих експертів в загальне розрізняють дві основні категорії асоціативних машин: статичні (static structure) і динамічні структури (dynamic structure). У статичних структурах рішення різних експертів об'єднуються за допомогою деякого механізму, який не враховує

вхідний сигнал. Ця категорія структур працює на основі наступних методів.

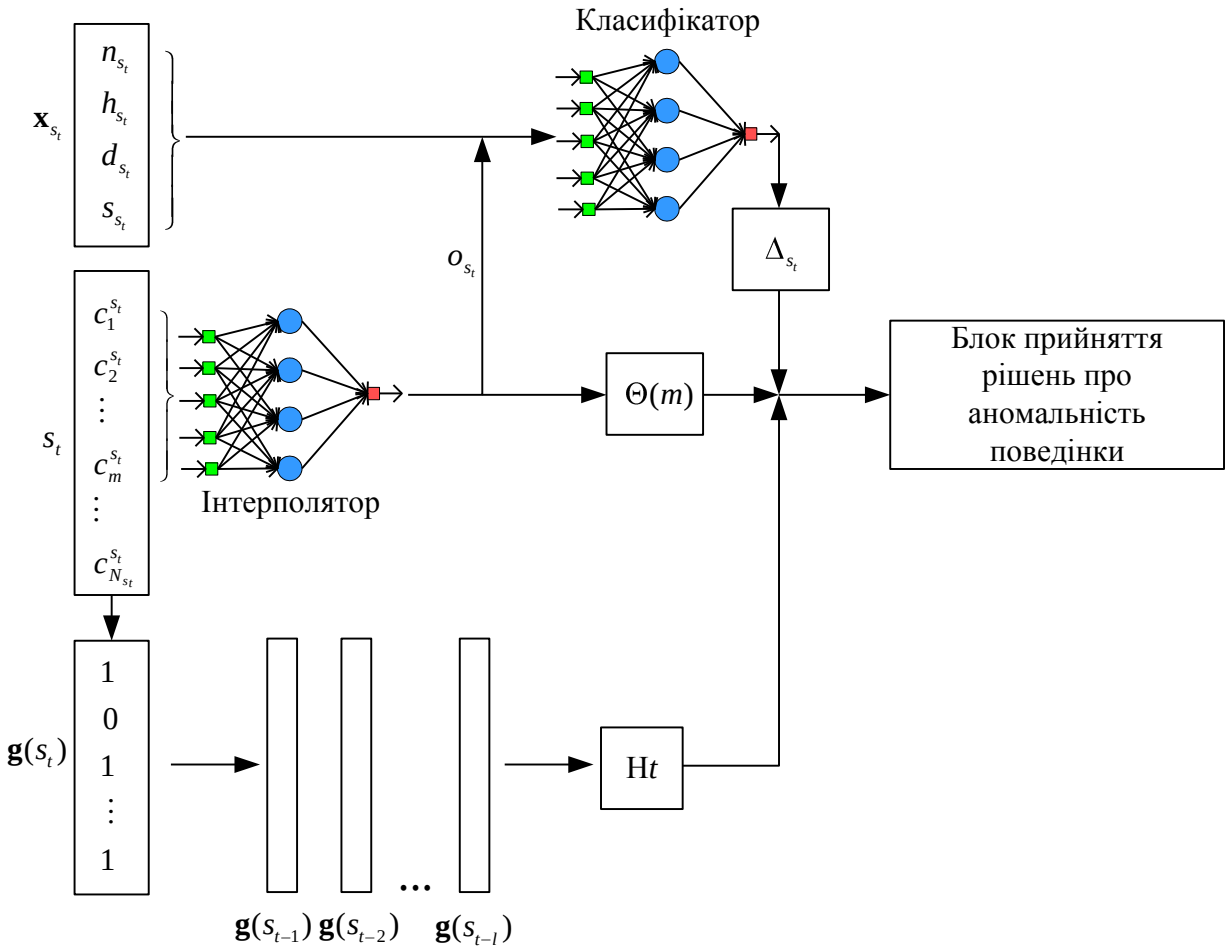


Рисунок 2.5 – Структура комплексної моделі

Усереднювання по ансамблю (ensemble averaging), при якому вихідний сигнал обчислюється як лінійна комбінація виходів окремих експертів.

Посилення (boosting), при якому експерти навчаються на різних підмножинах даних.

У свою чергу, в динамічних структурах (dynamic structure) асоціативних машин вхідний сигнал безпосередньо враховується в механізмі об'єднання вихідних сигналів експертів. Можна виділити дві різні реалізації динамічних структур:

1. Змішування думок експертів (mixture of experts), при якому думки окремих експертів об'єднуються в єдину шлюзову мережу (gating network).

2. Ієрархічне об'єднання думок експертів, при якому відгуки окремих експертів об'єднуються за допомогою декількох шлюзових мереж, організованих в ієрархічну структуру.

Асоціативна машина, яка використовується в даній роботі є динамічною структурою, що використовує методологію змішування думок експертів. В ролі експертів в даному випадку виступають нейромережеві моделі інтерактивної і сеансової складових і модуль аналізу трендів. Аналізуючи різні властивості поведінки користувача, кожен з експертів (нейронні мережі і модуль аналізу трендів) дає своє рішення про нормальну або аномальну поведінку користувача. За допомогою блоку ухвалення рішення асоціативна машина, в свою чергою, ухвалює загальне рішення.

Так, під час сеансу роботи користувача рішення про характер поведінки користувача ухвалюється на основі інтерактивної моделі користувача: якщо $\Theta(i)<\Theta'$ (де величина $\Theta(i)$ обчислюється згідно формули (2.3), Θ' – деякий поріг), то поведінка користувача вважається аномальною, інакше нормальною.

Після закінчення сеансу роботи користувача, окрім кількості вірно спрогнозованих команд за сеанс $\Theta(N_{s_t})$, є також результати роботи сеансової моделі і модуля моніторингу змін: Δ_{s_t} і H_t. Відповідно, загальне рішення про характер поведінки користувача ухвалюється з урахуванням всіх трьох величин. У даній моделі використовується лінійна комбінація величин:

$$\Omega=\alpha\Theta(N_{s_t})+\beta\Delta_{s_t}+\gamma(1-H_t),$$

де α, β, γ – коефіцієнти нормувань $\alpha+\beta+\gamma=1$, $\alpha, \beta, \gamma\in[0, 1]$.

Таким чином, виходячи з визначення величин $\Theta(N_{s_t})$, Δ_{s_t}, H_t, можна скласти наступне: чим менше значення Ω, тим більша ймовірність того, що поведінка користувача за сеанс була аномальною.

Коефіцієнтам α, β, γ можна також дати імовірнісну інтерпретацію: ці значення є ймовірністю правильної класифікації поведінки користувача за сеанс (нормальна/аномальна) (або ступінь довіри) відповідної складової комплексної моделі. Тому вибір значень α, β, γ можна здійснити, виходячи з таких міркувань: на основі тестових даних про роботу користувача за сеанс (які включають як приклади аномальної поведінки, так і нормальної) обчислюють відповідні частоти

правильної класифікації його поведінки кожної складової комплексної моделі. Тоді коефіцієнти α, β, γ слід вибирати пропорційними цим частотам. У даній же роботі всі три моделі однаково правильно відносили поведінки користувача до класу нормального або аномального, тому всі коефіцієнти α, β, γ вибиралися рівними 1/3. Проте питання вибору коефіцієнтів заслуговує уважного розгляду в рамках окремого дослідження, яке виходить за межі поставлених в даній роботі задач.

2.3 Генеративна модель поведінки користувачів комп'ютерних систем

2.3.1 Постановка задачі побудови генеративної моделі поведінки користувачів комп'ютерних систем

У загальному випадку функціонування нейронної мережі значно залежить від якості навчальної вибірки. Справа в тому, що при невеликому розмірі навчальної множини нейронна мережа має тенденцію до жорсткого запам'ятовування образів, що приводить до зменшення її властивості до узагальнення. Так, при побудові інтерактивної складової в нашому випадку проблема з навчальною вибіркою даних не виникала, оскільки навіть за нетривалий період часу роботи користувача може бути зібрана достатня кількість образів для навчання нейронної мережі. Наприклад, з урахуванням того, що користувач в середньому вводить від 80 до 150 команд за сеанс, то за десять сеансів навчальна вибірка може налічувати до 1000 образів.

У разі сеансової складової нейромережевою моделлю є класифікатор. Тому для стійкого розділення класів нормальної і аномальної поведінки потрібно забезпечити навчальну множину достатньо великого розміру. Проте для сеансової складової складно забезпечити навчальну вибірку даних, оскільки вхідними даними в даному випадку служить узагальнена інформація про сеанс роботи користувача. Тобто, на відміну від інтерактивної складової, в сеансовій кожен сеанс роботи користувача визначає всього один навчальний образ (або точку в просторі ознак). Так, за три місяці число таких сеансів може досягати 100, що в нашому випадку недостатньо для якісного навчання нейронної мережі і

оптимізації її архітектури. Для вирішення цієї проблеми можна використовувати два підходи. Перший з них полягає в значному збільшенні відрізку часу, в рамках якого відбувається спостереження за поведінкою користувача (скажемо до 8-10 місяців). Проте в цьому випадку велика ймовірність того, що за цей проміжок часу вона зміниться і, таким чином, навчальна множина міститиме суперечливі образи. Другий підхід полягає в статистичному моделюванні даних на основі наявної вибірки (побудова так званої генеративної моделі поведінки користувача). Він і буде використаний в даній дипломній роботі.

Оскільки в сеансовій складовій для навчання нейронної мережі використовується інформація про кількість команд, що вводяться, за сеанс, номері комп'ютера, тривалості і часі початку сеансу, для кожного користувача необхідно промоделювати саме цей набір даних.

Нехай є дані про роботу користувача за декілька місяців. Задача полягає в тому, щоб на основі цих даних висунути гіпотезу про їх імовірнісні характеристики, а потім перевірити ступінь відповідності цієї гіпотези реальним даним. В якості критерію згоди використовуватимемо так званий критерій χ^2 Пірсона [51].

Розглянемо принципи побудови критерію χ^2 і отримані залежності детальніше.

Нехай для кожного сеансу s_t є наступний набір даних про роботу користувача: $n_{s_t}, h_{s_t}, d_{s_t}, s_{s_t}\ {}_{t=1}^{T}$. Кожну з цих величин можна розглядати як випадкову, яка приймає T певних значень. До того ж, вважатимемо ці величини незалежними. Позначимо за допомогою X одну з цих випадкових величин. Розіб'ємо її значення по k інтервалах і представимо у вигляді наступного статистичного ряду:

I_i	$x_1; x_2$	$x_2; x_3$	…	$x_k; x_{k+1}$
p_i^*	p_1^*	p_2^*	⋯	p_k^*

де I_i – i-ий інтервал значень;

p_i^* – частота попадання в інтервал I_i.

Потрібно перевірити, чи узгоджуються ці дані з гіпотезою про те, що випадкова величина X має закон розподілу, заданий функцією розподілу $F(x)$ або щільністю $f(x)$ (назвемо її теоретичною).

Знаючи теоретичний закон розподілу, можна знайти теоретичну ймовріність попадання випадкової величини в той або інший інтервал:

$$p_1, p_2, \ldots, p_k.$$

Перевіряючи узгодженість теоретичного і статистичного (емпіричного) розподілів, виникає питання про те, яким же способом слід вибирати міру розбіжностей. В якості такої міри в математичній статистиці зазвичай вибирають суму квадратів відхилень $p_i^* - p_i$, взятих з деякими вагами c_i:

$$U = \sum_{i=1}^{k} c_i \left(p_i^* - p_i \right)^2.$$

Коефіцієнти c_i вводяться тому, що відхилення, що відносяться до різних інтервалів, не можна вважати рівноправними по значущості. Дійсно, одне і те ж по абсолютній величині відхилення $p_i^* - p_i$ може бути малозначним, якщо сама ймовірність p_i велика, і дуже помітна, якщо вона мала. Тому природно вибирати ваги c_i обернено пропорційними ймовірності p_i. Так, Пірсон показав, що якщо покласти

$$c_i = \frac{n}{p_i}, \tag{2.8}$$

то при достатньо великих значеннях n закон розподілу величини U практично не залежатиме від функції розподілу $F(x)$ і числа n, а залежатиме тільки від кількості інтервалів k і із збільшенням n

наближатися до розподілу χ^2 з щільністю:

$$f(u) = \frac{1}{2^{\frac{r}{2}}\Gamma\left(\frac{r}{2}\right)} u^{\frac{r}{2}-1} \mathrm{e}^{-\frac{u}{2}} \quad (u > 0)$$

де $\Gamma(\alpha) = \int_0^\infty t^{\alpha-1}\mathrm{e}^{-t}dt$ – гамма-функція;

r – кількість мір свободи.

Таким чином, міра розбіжності матиме такий вигляд:

$$\chi^2 \equiv U = \sum_{i=1}^{k} n \frac{p_i^* - p_i}{p_i}^2 . \qquad (2.9)$$

Оскільки розподіл χ^2 залежить від параметра r (число мір свободи), то для його обчислення використовують наступне правило:

$$r = k\text{-}s$$

де s – кількість незалежних умов, накладених на ймовірності p_i^*.

До прикладів таких умов можна віднести вимогу, щоб сума всіх частот p_i^* була рівна одиниці, або вимога рівності теоретичного середнього і дисперсії статичному і т. д.

Для розподілу χ^2 складені спеціальні таблиці. Користуючись ними, можна для кожного значення χ^2 і числа мір свободи r знайти ймовірність p того, що величина, розподілена за законом χ^2, перевершить це значення.

Таким чином, схема застосування критерію χ^2 до оцінки узгодженості статистичного і теоретичного розподілів зводиться до наступного:

а) Визначається міра розбіжності χ^2 відповідно до формули (2.9).

б) Визначається число мір свободи r як різниця між числом інтервалів k і

кількістю накладених зв'язків *s*.

в) По набутих значеннях χ^2 і r (за допомогою спеціальних таблиць для χ^2) визначається ймовірність p. Якщо ця ймовірність вельми мала, гіпотеза про відповідність статистичного розподілу теоретичному відкидається. Якщо ж ця ймовірність відносно велика, дану гіпотезу можна визнати такою, що не суперечить досвідченим даним.

2.3.2 Генерування значень випадкової величини із заданим законом розподілу

Припустимо тепер, що за допомогою критерію χ^2 вдалося визначити теоретичні розподіли для параметрів сеансової моделі. Тепер на їх основі необхідно згенерувати відповідні набори значень для заданої кількості сеансів. Ця задача зводиться до задачі генерування значень випадкової величини, розподіленої по заданому закону. Скористаємося для вирішення задачі наступним твердженням, що дозволяє генерувати значення випадкової величини із заданим безперервним розподілом [51].

Нехай ξ – випадкова величина з безперервною функцією розподілу F_ξ. Тоді випадкова величина $\eta=F_\xi(\xi)$ буде рівномірно розподілена на відрізку [0; 1].

Таким чином, для генерування значень із заданим розподілом F, необхідно спочатку згенерувати випадкову величину η, розподілену рівномірно на відрізку [0; 1], а потім підставити набутих значень у функцію, зворотну до функції розподілу F, тобто $F^{-1}(\eta)$.

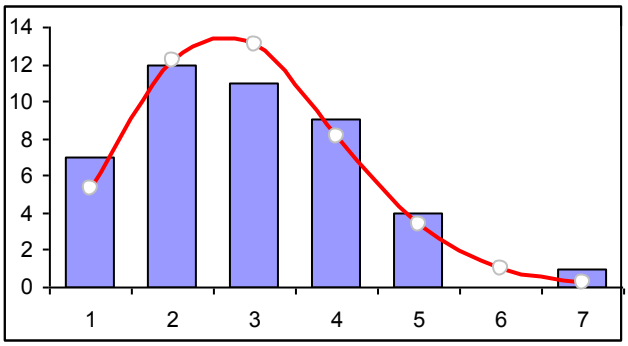
2.3.3 Апроксимація емпіричних залежностей теоретичними

Розглянемо тепер залежності, які були отримані при моделюванні. В якості теоретичних розподілів використовувалися рівномірний, нормальний і логарифмічно нормальний.

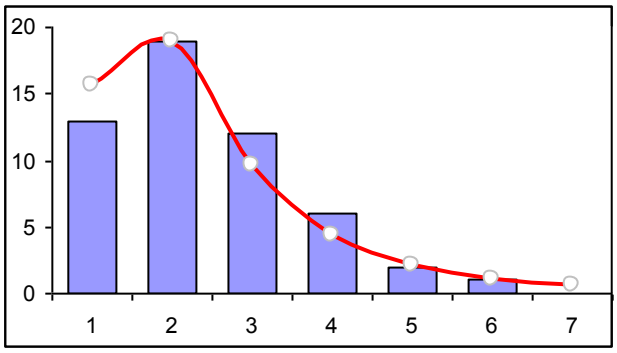
Кількість команд, що вводяться.

Для різних користувачів був проведений аналіз різних розподілів. Якнайкраще значення критерію χ^2, в середньому рівне 2,1, було отримано для логарифмічного нормального розподілу. Оскільки в даному випадку кількість мір

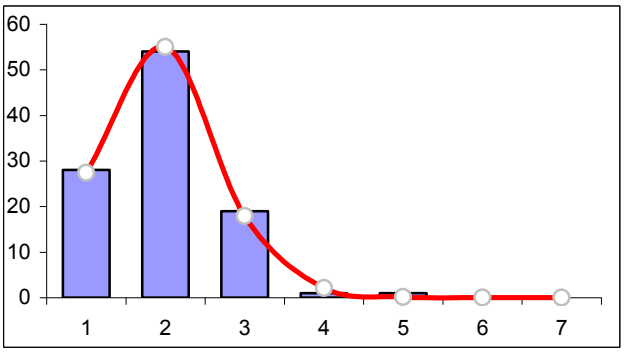
свободи r дорівнювала 4 ($k = 7$, $s = 3$), це значення забезпечувалj 72% відповідність гіпотези реальним даним. На рисунку 2.6 приведений емпіричний і побудований теоретичний розподіл для цього параметра для різних користувачів.



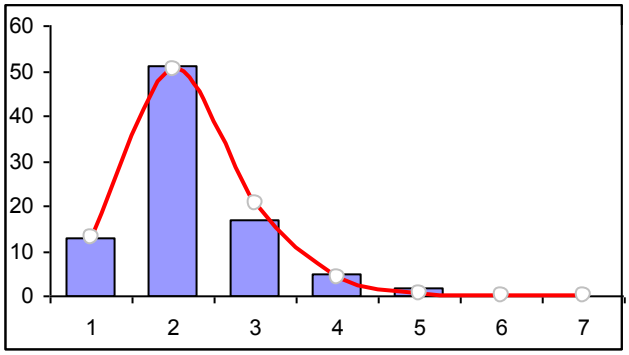
користувач № 1



користувач № 2



користувач № 3

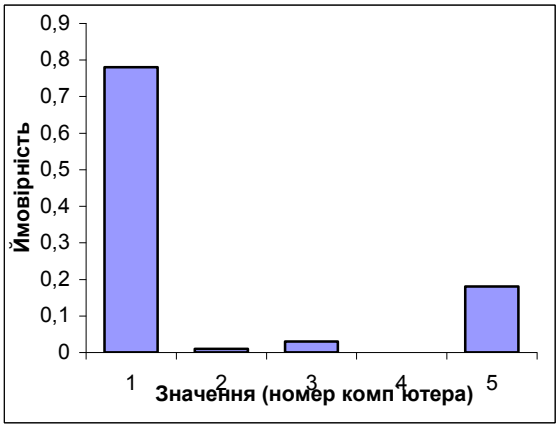


користувач № 4

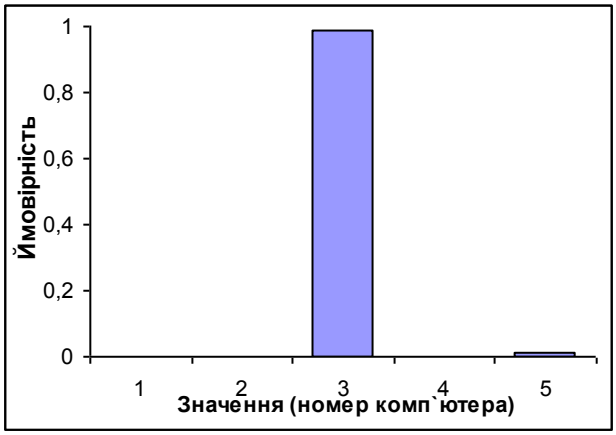
Рисунок 2.6 – Емпіричний (гістограма) і теоретичний (суцільна лінія) розподіли для кількості команд, що вводяться, для різних користувачів

Номер комп'ютера.

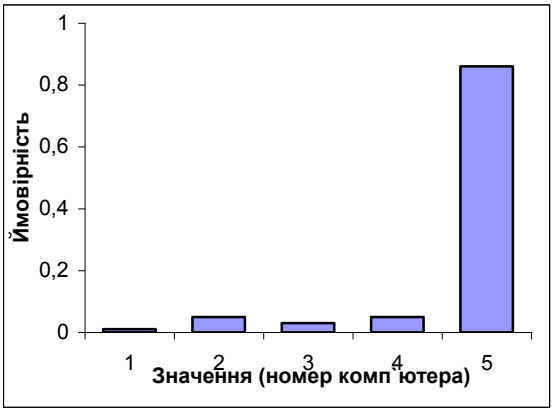
При аналізі значень цього параметра необхідно враховувати наступне: користувач, як правило, має своє основне місце роботи за комп'ютером і дуже рідко працює за іншими. Тому завжди існує значення, ймовірність якого найбільша і складає 0,75...0,95 (рисунок 2.7). Події ж, пов'язані з роботою користувача за іншими робочими станціями, можна вважати рівноімовірними.



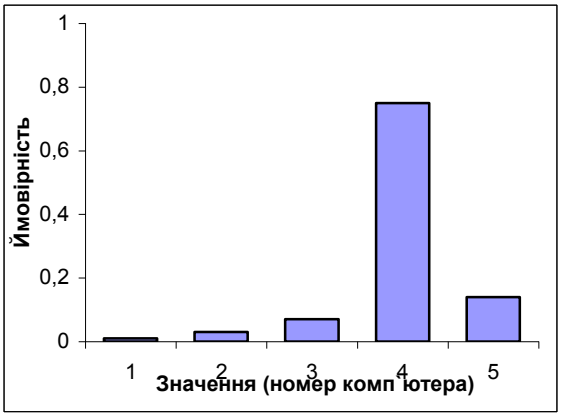
користувач № 1



користувач № 2



користувач № 3



користувач № 4

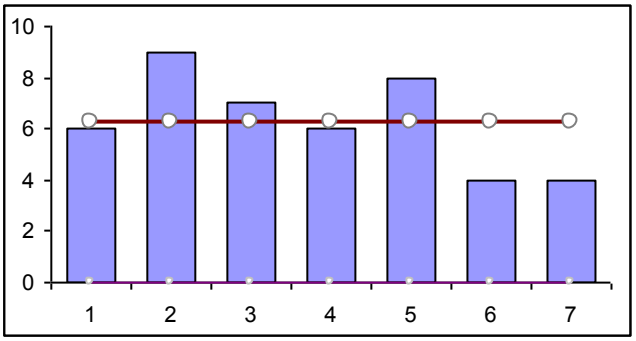
Рисунок 2.7 – Розподіл по номерах комп'ютерів для різних користувачів

Тривалість сеансу.

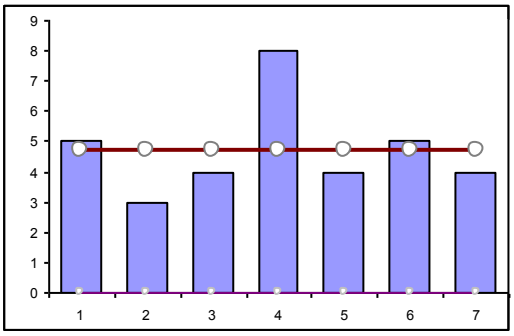
Аналіз значень цього параметра показує, що вони розподілені рівноімовірно (рисунок 2.8). Значення критерію χ^2, в середньому рівне 2,19, (при значеннях параметрів $r = 4$, $k = 7$, $s = 3$) забезпечує в середньому 70% відповідність гіпотези реальним даним.

Час початку сеансу.

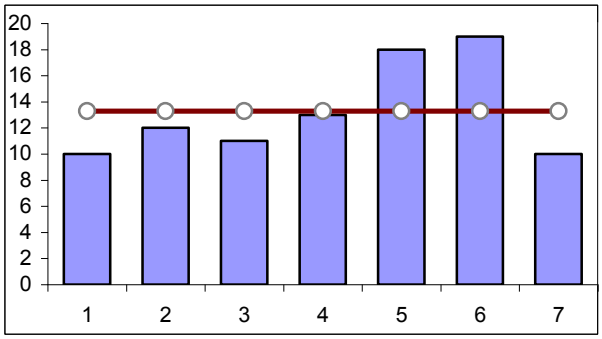
Якнайкраще значення критерію χ^2, в середньому рівне 1,08, для цього параметра було отримано для нормального розподілу. При $r = 4$, $k = 7$, $s = 3$ це значення в середньому забезпечувало 89% відповідність гіпотези реальним даним. На рисунку 2.9 приведений емпіричний і побудований теоретичний розподіл.



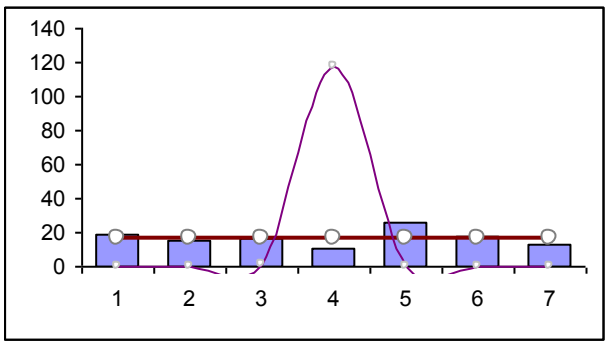
користувач № 1



користувач № 2

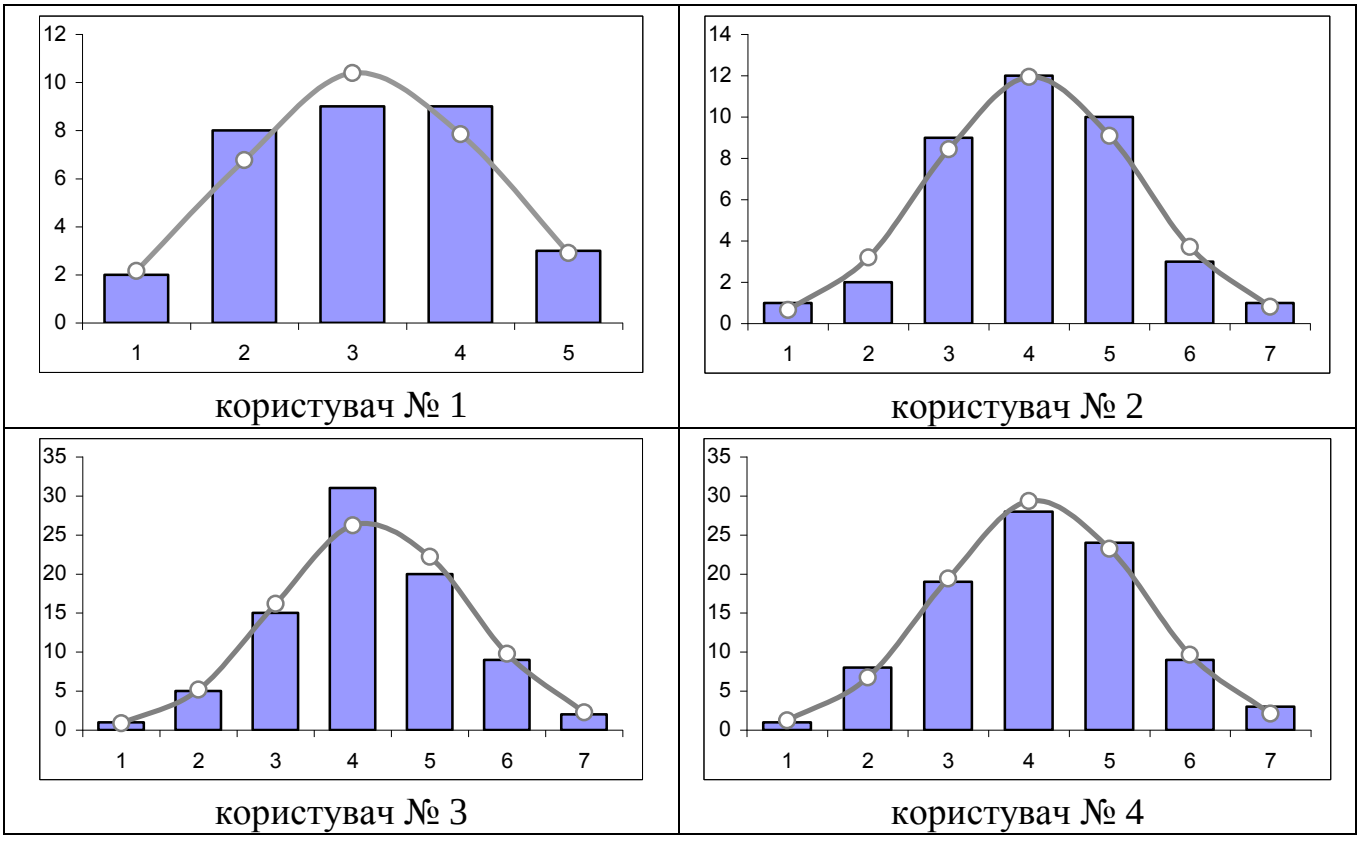


користувач № 3



користувач № 4

Рисунок 2.8 – Емпіричний (гістрограма) і теоретичний (суцільна лінія) розподіли для тривалості сеансу для різних користувачів



користувач № 1

користувач № 2

користувач № 3

користувач № 4

Рисунок 2.9 – Емпіричний (гістрограма) і теоретичний (суцільна лінія) розподіли для часу початку сеансу для різних користувачів

2.3.4 Реалізація генеративної моделі

На основі отриманих теоретичних залежностей була розроблена програма, що моделює роботу користувачів комп'ютерних систем за сеанс. Призначений для користувача інтерфейс даної програми (рисунок 2.10, а) дозволяє вводити кількість сеансів, за які необхідно згенерувати дані, загальне число комп'ютерів і параметри розподілів складових сеансової моделі:

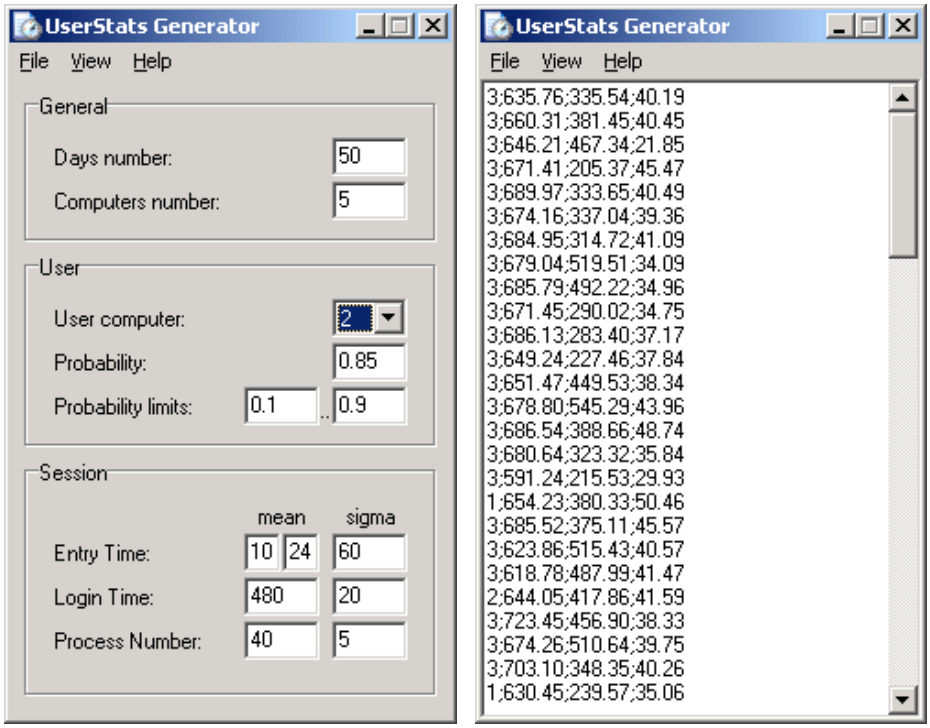
– для тривалості сеансу – середнє і дисперсію для логарифмічного нормального розподілу;

– для номера комп'ютера – ймовірність комп'ютера, за яким користувач працює найчастіше;

– для тривалості сеансу – межі рівномірного розподілу;

– для часу початку сеансу – середнє і дисперсію для нормального розподілу.

Дана програма дозволяє генерувати навчальну множину для сеансової моделі користувача. Імовірнісні характеристики згенерованих даних відповідають статистичним показникам, отриманим на реальних даних. Приклад роботи програми показаний на рисунку 2.10, б.



(а) (б)

Рисунок 2.10 – Інтерфейс системи генерування даних про роботу користувачів комп'ютерних систем за сеанс (а) і результати її роботи (б)

Згенеровані за допомогою програми дані дозволяють оптимізувати структуру нейромережевої моделі і покращити якість функціонування сеансової складової комплексної моделі поведінки користувачів.

У даному розділі запропонована комплексна модель поведінки користувачів комп'ютерних систем, яка містить три компоненти: інтерактивну (прогнозну) складову, сеансову (статистичну) складову і модуль аналізу трендів.

Інтерактивна складова грунтується на прогнозуванні дій користувача на основі попередніх, що дозволяє описати динамічні властивості його поведінки. Для прогнозування використовується нейромережевий підхід. Це пояснюється тим, що поведінкою користувача є нелінійний динамічний процес, а також необхідністю прогнозування в режимі реального часу на основі реальних даних. У основу інтерактивної моделі поведінки користувачів покладена нейронна мережа прямого розповсюдження. На основі кількості команд, які були правильно спрогнозовані нейронною мережею, робиться висновок про те, чи відповідає поточна поведінка користувача раніше побудованій моделі (тобто чи є поведінка нормальною або аномальною).

Запропонована сеансова складова враховує статистичні параметри поведінки користувача (сигнатура користувача), які були отримані впродовж сеансу, а саме: кількість введених команд; результати інтерактивної моделі; набір комп'ютерів; тривалість сеансу; час початку сеансу. Цей набір даних використовується в якості вхідних даних для виявлення нормальної або аномальної роботи користувача за сеанс в цілому. Як і у випадку інтерактивної моделі, для реалізації моделі використовується нейронна мережа прямого розповсюдження. В даному випадку нейронна мережа працює як класифікатор.

Слід виділити такі переваги нейромережевих складових комплексної моделі:

– незалежність від кількості користувачів в системі;

– можливість виявлення прихованих закономірностей в поведінці користувача;

– адаптація до зміни поведінки користувачів.

Оскільки користувачам комп'ютерних систем властиво змінювати свою

поведінку з часом, для адаптації до цього нейромережевих складових важливо забезпечити модуль, який виявлятиме ці зміни. У даній роботі запропоновано використовувати модуль аналізу трендів, який дозволяє виявляти «тренди» поведінки і надає додаткове підтвердження аномальності поведінки користувачів. Його функціонування засноване на використанні інформації про команди, що вводяться за сеанс і їх порівняння з даними попередніх сеансів. Таким чином, використання модуля аналізу трендів дозволяє відрізнити природну зміну поведінки користувача від аномальної, а також забезпечити адаптацію нейромережевих складових до можливих змін.

Також розроблена генеративна модель поведінки користувачів, що забезпечує репрезентативні набори даних для верифікації і ідентифікації комплексної моделі. Для побудови даної моделі виконувалося статистичне моделювання параметрів сеансової складової для кожного користувача: кількості команд, що вводяться, за сеанс, номери комп'ютера, тривалості і часу початку сеансу. Емпіричні розподіли були апроксимовані теоретичними. В якості критерію згоди використовувався критерій χ^2 Пірсона.

3 РЕАЛІЗАЦІЯ НЕЙРОМЕРЕЖЕВОЇ МОДЕЛІ ПОВЕДІНКИ КОРИСТУВАЧІВ КОМП'ЮТЕРНИХ СИСТЕМ НА ОСНОВІ АГЕНТНОЇ ТЕХНОЛОГІЇ

При реалізації комплексної нейромережевої моделі користувачів комп'ютерних систем необхідно враховувати наступні чинники: як правило, комп'ютерна система є розподіленою і гетерогенною і містить велику кількість користувачів, для кожного з яких будується своя модель. При цьому з метою зменшення навантаження на комп'ютерну систему модель поведінки доцільно реалізовувати як автономний модуль, що має також можливість «переміщатися» в системі, оскільки користувач може працювати за різними робочими станціями. Всі ці чинники визначають ті властивості, якими повинна володіти система моніторингу діяльності користувачів, що розробляється. До них відносяться розподіленість, масштабованість, можливість роботи з різними операційними системами і форматами даних, автономність і мобільність компонентів (модулів) і т. д. Цим вимогам краще всього задовольняє агентна парадигма реалізації розподілених систем [52, 53].

3.1 Агентна технологія

У даному розділі агентний підхід використовується для реалізації системи моніторингу діяльності на основі комплексної моделі користувача, описаної в розділі 2. Застосування агентної технології для реалізації даної системи обумовлене наступним: необхідність створення великої кількості моделей (для кожного користувача потрібно побудувати дві відповідні нейромережеві моделі), необхідність мобільності (агент переміщається на клієнтський комп'ютер), взаємодія з базою даних для навчання.

У даній роботі в якості базового визначення агента виберемо наступне: агент – це сутність, яка може приймати інформацію із зовнішнього середовища і реагувати на зовнішні впливи [52]. Введемо формальний опис агента. У загальному випадку агент визначається набором:

<S, Prog, Eff, Arch, P, A, G, E>, (3.1)

де E (environment) – зовнішнє середовище, в якому функціонує агент;

S (sensors) – множина входів, за допомогою яких агент сприймає інформацію із зовнішнього середовища;

Eff (effectors) – множина виходів, за допомогою яких агент впливає на зовнішнє середовище;

P (percepts) – інформація, яку отримує агент;

A (actions) – реакція агента;

Prog (program) Prog: P–>A – функція, що визначає залежність реакції агента від вхідних дій;

G (goal) – цілі, які досягає агент;

Arch (architecture) – фізична оболонка, яка об'єднує всі базові елементи агента.

Набір S, Prog, Eff, Arch визначає базову конструкцію агента (його каркас), тоді як P, A, G, E його змістовне “наповнення”.

Більшість робіт виділяють наступні основні властивості, якими повинні володіти агенти [54, 55]: автономність, реактивність (здатність агента реагувати на зміни в зовнішньому середовищі за прийнятний час), активність (здатність агента вирішувати поставлені задачі), здатність взаємодіяти (спілкуватися) з іншими агентами (можливо з людиною) за допомогою мови спілкування агентів (ACL Agent Communication Language), мобільність (здатність агентів переміщатися в зовнішньому середовищі), адаптивність (здатність агента до навчання).

У даній роботі для реалізації системи моніторингу використовуватимуться так звані програмні агенти (software agents). Їх відмінність полягає в тому, що вони є комп'ютерними програмами і функціонують в комп'ютерних системах. Для програмного агента (в термінах даного вище визначення (3.1)): *E=комп’ютерна система*, *Arch=програма* (код), а *S* і *Eff* – деякі функції (які в загальному випадку можуть також бути програмами), за допомогою яких агент обмінюється інформацією із зовнішнім середовищем.

3.2 Система моніторингу діяльності користувачів комп'ютерних систем на основі комплексної нейромережевої моделі

Оскільки комплексна модель користувача враховує як динамічні (інтерактивна складова), так і статистичні (сеансова складова і модуль аналізу трендів) властивості поведінки користувачів в даній системі моніторинг здійснюється як в режимі реального часу (on-line), так і в автономному режимі (off-line).

Метою моніторингу в режимі реального часу є виявлення аномальної діяльності під час роботи користувача на основі інтерактивної моделі. Автономний моніторинг призначений для виявлення нехарактерної діяльності користувача за сеанс в цілому і виявлення можливих трендів в його роботі. Для цього використовуються сеансова модель і модуль аналізу трендів.

3.2.1 Архітектура системи

Система моніторингу діяльності користувачів комп'ютерних систем містить наступні основні компоненти:

- агент, що інкапсулює (що містить в собі і приховує від інших об'єктів) інтерактивну модель користувача і що працює в режимі реального часу (On-line User Agent);
- агент, що інкапсулює сеансову модель користувача і модуль аналізу трендів і що працює в автономному режимі (Off-line User Agent);
- агент-контролер (Controller Agent), керівник роботою інших агентів в системі;
- база даних (Database), що містить дані і параметри існуючих моделей.

Розглянемо кожен з компонентів запропонованої системи моніторингу детальніше (рисунок 3.1).

Агент, що інкапсулює інтерактивну модель користувача, функціонує під час роботи користувача в комп'ютерній системі в режимі реального часу і дозволяє виявляти аномалії (відхилення від нормальної поведінки) в його діяльності.

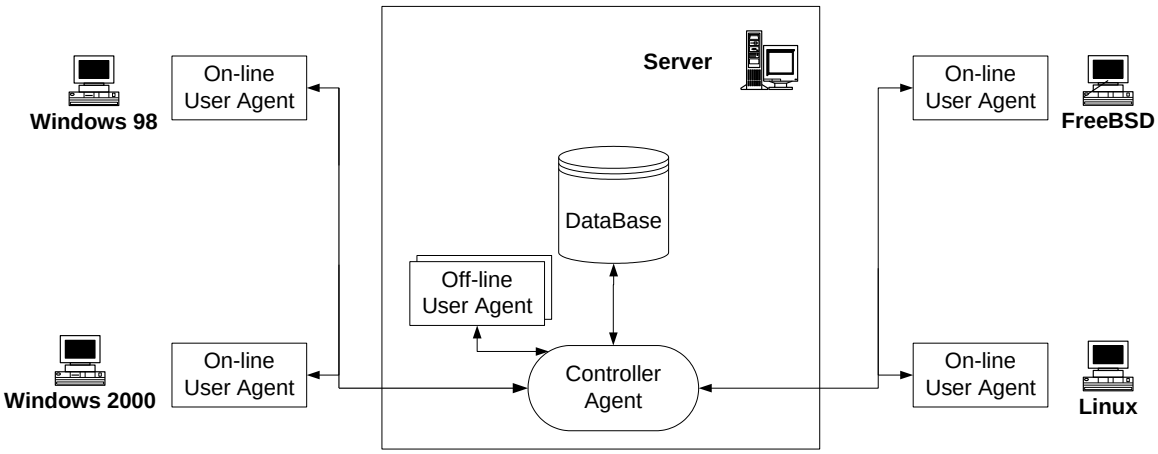


Рисунок 3.1 – Архітектура системи моніторингу

У позначеннях формули (3.1) цей тип агента характеризується наступними параметрами (з урахуванням того, що в нашому випадку параметри *E, Arch, S* і *Eff* для програмного агента фіксовані):

G – виявлення аномальної діяльності користувача під час його роботи;

P – послідовність команд, які введені користувачем;

A – прогнозована команда;

Prog: P–>A – збір даних про діяльність користувача, прогнозування команд з використанням нейронної мережі і обчислення значення $\Theta(i)$.

В процесі роботи користувача даний агент прогнозує його дії, які згодом порівнюються з реальними. Якщо відносна кількість правильно спрогнозованих команд впродовж сеансу більша заданого порогу, вважається, що поведінка користувача нормальна, інакше аномальна. Ще одна функція цього агента полягає в зборі інформації про поведінку користувача і її збереження в базі даних. При цьому даний тип агента повинен бути розроблений для різних операційних систем (ОС), які використовуються в комп'ютерній системі (наприклад, Win2000/XP, Win98, FREEBSD).

Агент, що інкапсулює сеансову модель користувача, функціонує після завершення сеансу роботи користувача і має наступні значення параметрів набору, який визначається формулою (3.1):

G – виявлення нехарактерної діяльності користувача за сеанс в цілому і

виявлення можливих трендів в його роботі;

P – статистичні дані про діяльність користувача;

A – ймовірність характерної діяльності користувача;

Prog: P–>A – обчислення ймовірності Δ_{s_t} нехарактерної діяльності користувача з використанням нейронної мережі.

На основі зібраної агентом, що працював в режимі реального часу, інформації агент, що інкапсулює сеансову модель користувача, з'ясовує, на скільки діяльність користувача була аномальною (що визначається числом з проміжку [0, 1], яке вказує ймовірність нормальної поведінки користувача).

Агент-контролер – агент, який відповідає за функціонування системи в цілому і управляє роботою інших агентів. У позначеннях формули (3.1) цей тип агента має наступний опис:

G – управління системою моніторингу в цілому;

P – дані про різні елементи системи;

A – організація взаємодії між елементами системи;

Prog: P–>A – створення різних типів агентів, отримання інформації з бази даних про параметри моделі, перенавчання нейронних мереж (тобто адаптація моделі).

Цей агент отримує інтегральні дані про поведінку всіх елементів системи. З його допомогою організовується взаємодія в рамках системи і контролюється її функціонування, а також відбувається спілкування між агентами. Він звертається до бази даних з метою отримання і передачі інформації. Під його контролем відбувається навчання різних типів агентів. Він також поповнює базу даних новими даними про користувачів комп'ютерної системи.

У базі даних зберігається інформації, яка необхідна для функціонування системи.

3.2.2 Основні принципи функціонування системи

Коли користувач (`User`) входить в комп'ютерну систему (тобто починає роботу за комп'ютером Host), агент-контролер (Controller Agent) створює і ініціалізовує відповідного агента, що інкапсулює інтерактивну модель

користувача (On-line User Agent). Отримавши параметри моделі користувача з бази даних (DB) (тобто параметри нейронної мережі для даного користувача), агент переміщається на сторону клієнта і починає свою роботу, виконуючи свої функції по прогнозуванню команд і збору інформації з файлу-аудиту (Logger). Якщо виявлена аномальна або нехарактерна робота користувача, відповідне повідомлення посилається агентові-контролеру. Після завершення сеансу користувача даний агент знищується. Діаграма послідовностей для агента даного типу приведена на рисунку 3.2.

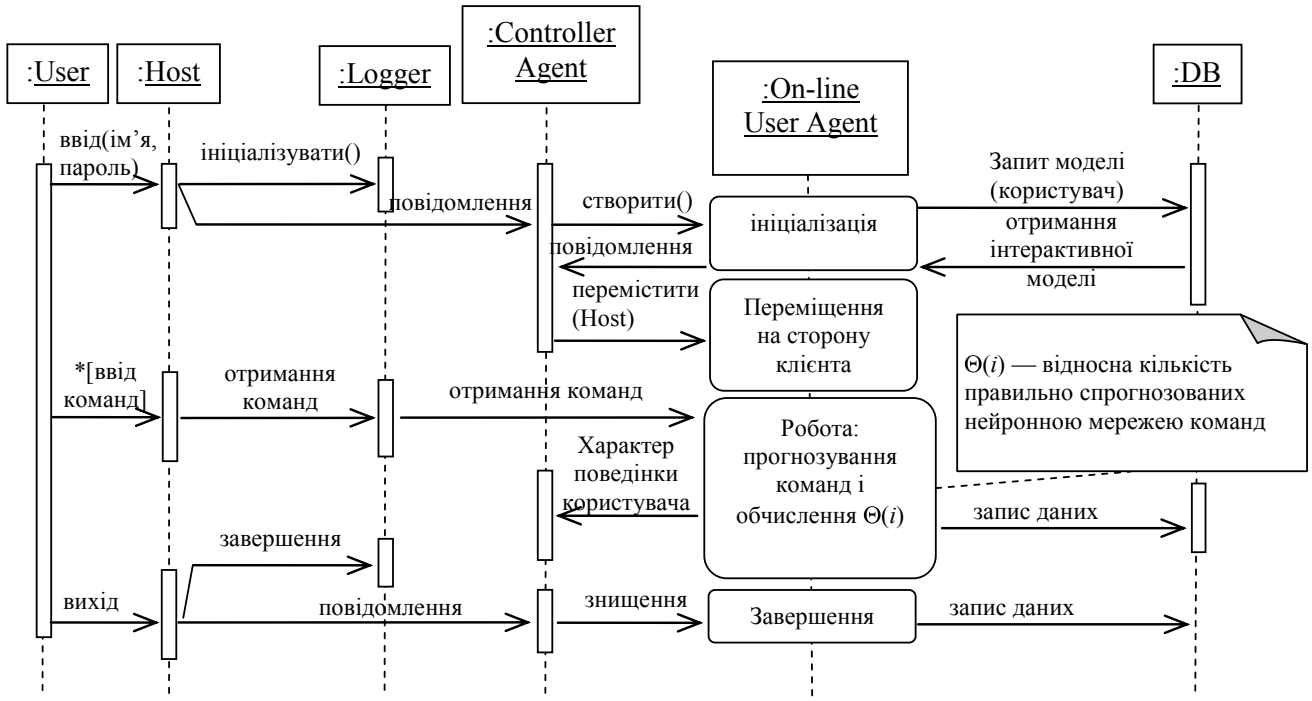


Рисунок 3.2 – UML–діаграма послідовностей, що відображає роботу агента, який інкапсулює інтерактивну модель

Після закінчення робочого дня (коли навантаження на систему мінімальне) агент-контролер створює і ініціалізовує відповідного агента, що інкапсулює сеансову модель користувача (Off-line User Agent). На основі статистичних даних, які були отримані під час сеансу користувача, сеансової моделі і модуля аналізу трендів цей агент визначає характер активності користувача (нормальна або аномальна) і можливі тренди в його роботі. У разі аномальної активності відповідна інформація посилається агентові-контролеру. Діаграма послідовностей для даного агента представлена на рисунку 3.3.

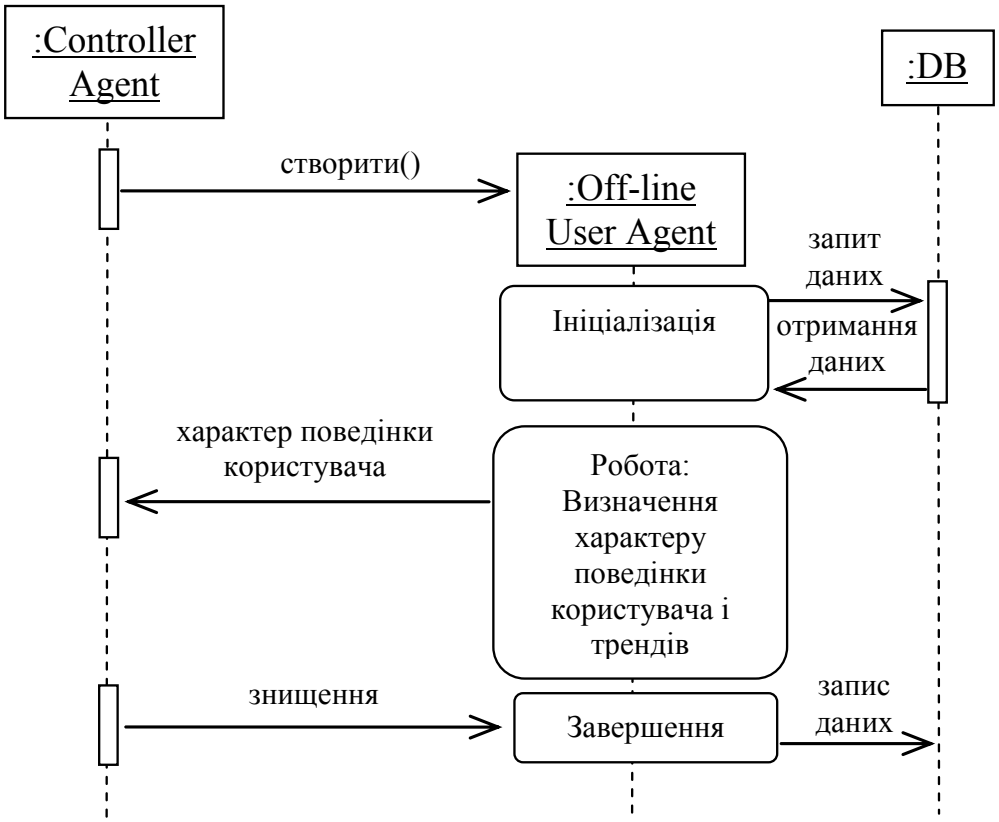


Рисунок 3.3 – UML–діаграма послідовностей, що відображає роботу агента, який інкапсулює сеансову модель

3.2.3 Діаграма класів для реалізації нейромережевих моделей

Класи для реалізації нейромережевих моделей згруповані в пакет `neural`. Відповідна діаграма приведена на рисунку 3.4.

Клас `UserAgent` реалізує шаблон інтелектуального агента, що інкапсулює нейромережеві моделі. Клас Layer пакету neural реалізує шар нейронної мережі. Атрибути LSize, Weights і Outputs даного класу визначають відповідно кількість нейронів в шарі, значення вагових коефіцієнтів і вихід нейронів шару. Метод dTrFuntion() реалізує активаційну функцію нейронів, а метод Run() призначений для обчислення виходу нейронів.

Клас `TestNeural` призначений для реалізації режиму прогону нейронної мережі. Даний клас використовує масив об'єктів типу Layer для реалізації нейронної мережі. Атрибути Lnum і pdWeights задають кількість шарів в нейронній мережі і матрицю вагових коефіцієнтів. Метод `Test()` обчислює вихід нейронної мережі при подачі вхідного вектора.

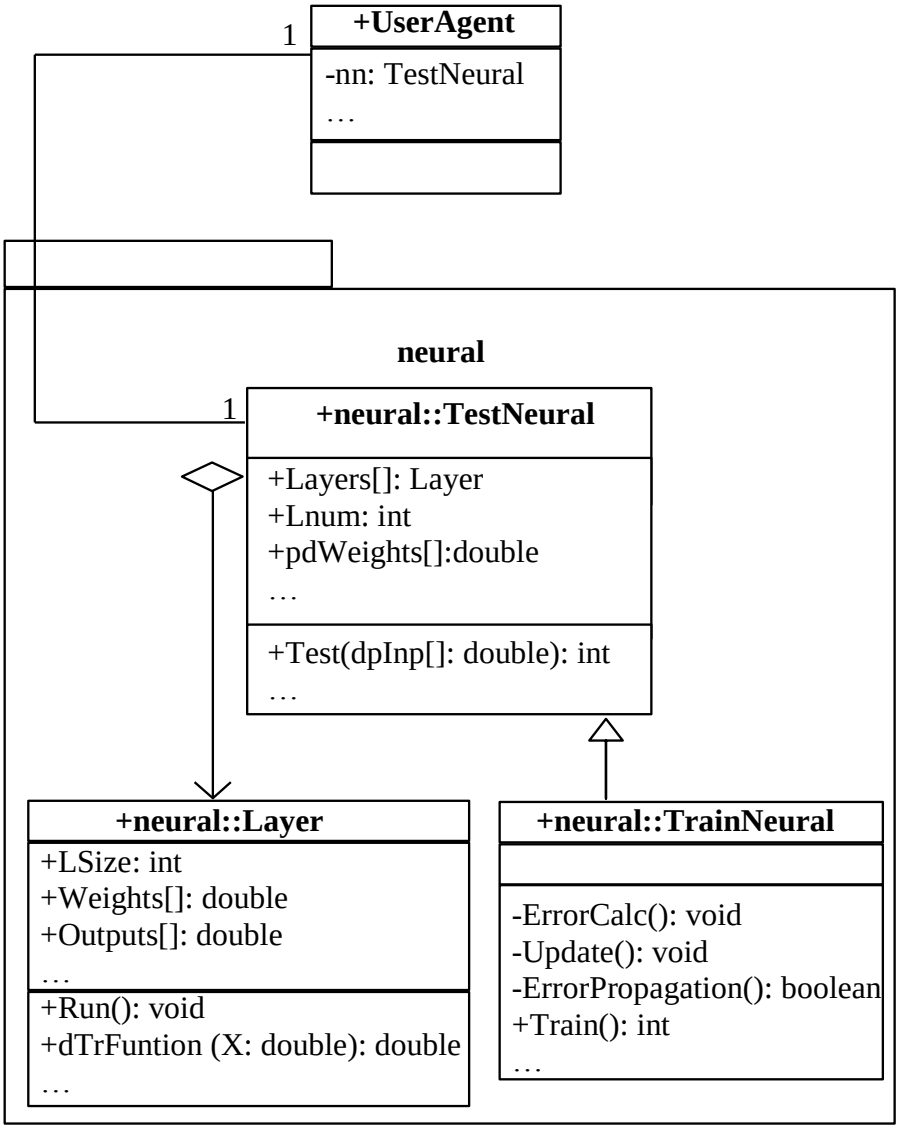


Рисунок 3.4 – UML–діаграма класів, що реалізовують нейромережеві моделі

У класі `TrainNeural` реалізовані методи, призначені для навчання нейронної мережі. Метод ErrorCalc() використовується для обчислення середньоквадратичної помилки, а метод Update() для зміни вагових коефіцієнтів. Метод ErrorPropagation() реалізує алгоритм зворотного розповсюдження помилки. Метод Train() є основним, за допомогою якого здійснюється ітераційна зміна вагових коефіцієнтів методом градієнтного спуску.

Програмний код реалізації класів пакету `neural` приведений в додатку А.

3.3 Реалізація системи нейромережевого моніторингу

3.3.1 Засоби програмної реалізації

Система нейромережевого моніторингу активності користувачів комп'ютерних системах була реалізована на основі мобільних агентів (вид агентів, які здатні переміщатися в мережі). В якості технології реалізації і середовища програмування мобільних агентів були вибрані, відповідно, Java і Aglets Software Development Kit (ASDK).

Java як мова програмування мобільних агентів надає широкий набір можливостей, які дозволяють спростити розробку багатоагентних систем. Слід зазначити такі властивості Java, як платформна незалежність, безпечне виконання кодів, динамічне завантаження класів, багатопотокове програмування, серіалізація об'єктів (тобто уявлення у вигляді бінарної послідовності даних). В той же час, слід виділити такі недоліки даної технології, які пов'язані з програмуванням мобільних агентів на Java: неадекватна підтримка контролю ресурсів, відсутність захисту посилань на об'єкти, відсутність підтримки збереження і відновлення стану виконання.

ASDK є програмним забезпеченням, яке вільно поширюється компанією IBM і призначене для створення мобільних агентів, які в реалізації ASDK називаються аглетами. Версія 2.0.2 [56] має просту і прозору структуру, зручний графічний інтерфейс користувача (що надається сервером аглетів Tahiti) і корисну документацію. Варто виділити наступні властивості ASDK [57]:

− використання спеціально розробленого стандарту MASIF (Mobile Agent System Interoperability Facility) для забезпечення взаємодії між різними агентними системами;

− використання на прикладному рівні протоколу ATP (Aglets Transfer Protocol) з підтримкою HTTP-тунелювання (HTTP Tunneling) для переміщення агентів;

− мобільність агентів, яка заснована на серіалізації і переміщенні байт-коду Java з одного місця в інше;

− використання стандартної системи безпеки Java (JDK keytool).

Розглянемо тепер основні принципи створення аглетів з використанням ASDK.

3.3.2 Основні принципи створення аглетів

У загальному випадку аглети – це об'єкти Java, які можуть переміщатися з одного комп'ютера мережі на інший. Під переміщенням аглета розуміють передачу програмного коду із збереженням стану агента (тобто значень атрибутів). Таким чином, якщо агент виконує деякі дії на одному комп'ютері, то у будь-який момент він може зупинитися, переміститься на інший і продовжити там свою роботу.

Базова функціональність мобільних агентів визначається програмним інтерфейсом Aglet API (Application Programming Interface). На рисунку 3.5 показані основні інтерфейси і класи Aglet API, а також взаємозв'язки між ними.

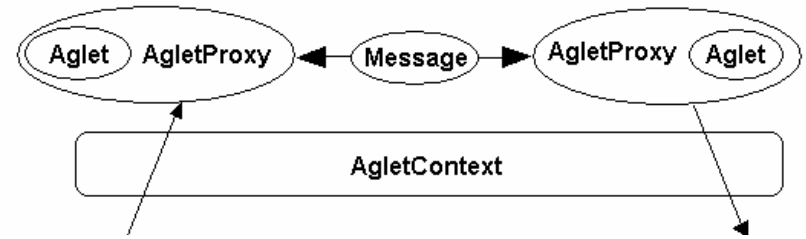


Рисунок 3.5 – Основні інтерфейси і класи Aglet API

Опишемо основне призначення інтерфейсів і класів інтерфейсу Aglet API:

Клас `com.ibm.aglet.Aglet` базовий, він визначає основні атрибути і методи, які повинен містити мобільний агент. При цьому будь-який створюваний аглет повинен успадковувати цей абстрактний клас.

Інтерфейс `com.ibm.aglet.AgletProxy` призначений для створення об'єкту-посередника для аглета. Це означає, що будь-який запит, адресований агентові для виконання тих або інших дій, здійснюватиметься через його посередника, який на основі установок диспетчера безпеки (SecurityManager) визначає, які методи аглета можна викликати, а які ні.

Інтерфейс `com.ibm.aglet.AgletContext` призначений для створення оточення (runtime environment), в рамках якого існуватимуть агенти (тобто агент

не може існувати сам по собі).

Клас `com.ibm.aglet.Message` призначений для створення об'єктів, з використанням яких здійснюється спілкування між агентами.

Як вже наголошувалося, будь-який агент, що створюється користувачем, повинен успадковувати клас `com.ibm.aglet.Aglet` або одну з його реалізацій. Нижче приведені методи даного класу, які при цьому необхідно перевизначити:

`void onCreation(Object init)` – цей метод викликається один раз при створенні нового аглета;

`void run()` – цей метод викликається кожного разу при створенні, переміщенні, активації або клонуванні (тобто створенні ідентичної копії) аглета;

`void onDisposing()` – цей метод викликається при знищенні аглета.

У наступному фрагменті коду приведений приклад створення аглета:

```
     import com.ibm.aglet.*; // Підключення пакету аглетів
com.ibm.aglet

     public class UserAglet extends Aglet {
          public void onCreation(Object init){
          System.out.println("Aglet is created!");



          public void run() {
          System.out.println("Hello!");



          public void onDisposing() {
          System.out.println("bye!");
          }
     }
```

Щоб забезпечити мобільність аглетів (тобто можливість переміщатися на віддалений комп'ютер), в ASDK використовується модель подій (event model). Так, при відправці або прибутті агента на комп'ютер, генерується відповідна

подія, яка пов'язана з одним з методів інтерфейсу `com.ibm.aglet.event.MobilityListener` (саме він відповідає за обробку подій, пов'язаних з мобільністю аглетів). Створивши клас, що реалізовує цей інтерфейс, користувач може перевизначити потрібні методи. Розглянемо приклад:

```
     import com.ibm.aglet.Aglet;
     import com.ibm.aglet.event.MobilityEvent;
     import com.ibm.aglet.event.MobilityListener;
     class UserListener implements MobilityListener{ // Реалізація
інтерфейсу MobilityListener

          public void onDispatching(MobilityEvent event){
          // Цей метод викликається безпосередньо перед
переміщенням аглета
          }

          public void onArrival(MobilityEvent event){
          // Цей метод викликається безпосередньо після прибуття на
новий хост
          }

          public void onReverting(MobilityEvent event){
          // Цей метод викликається перед переміщенням аглета
«додому»
          }
     }

     public class UserAglet extends Aglet {
          public void onCreation(Object init){
          MobilityListener listener = new UserListener(); //
Створення екземпляра призначеного для користувача класу
UserListener
          addMobilityListener(listener); // Зв'язує обробник подій
з об'єктом listener
          }
     }
```

3.3.3 Реалізація і результати роботи системи моніторингу

Запропонована система моніторингу була реалізована на основі клієнт-серверної архітектури. Серверна частина є спеціальним додатком (серверною платформою), до функцій якого відносяться створення і розміщення агентів системи (всі агенти, які використовуються в системі, породжуються саме тут), перенаправлення запитів, взаємодія з базою даних. Клієнтська частина призначена для забезпечення роботи агентів на призначеному для користувача комп'ютері. Її функції полягають в розміщенні агентів на стороні користувача і ведення аудиту про його діяльність. Також був реалізований графічний інтерфейс (рисунок 3.6), в якому відображається інформація про ім'я користувача і використовувану операційну систему, про параметри клієнтської платформи, про стан агента і дані про роботу агента.

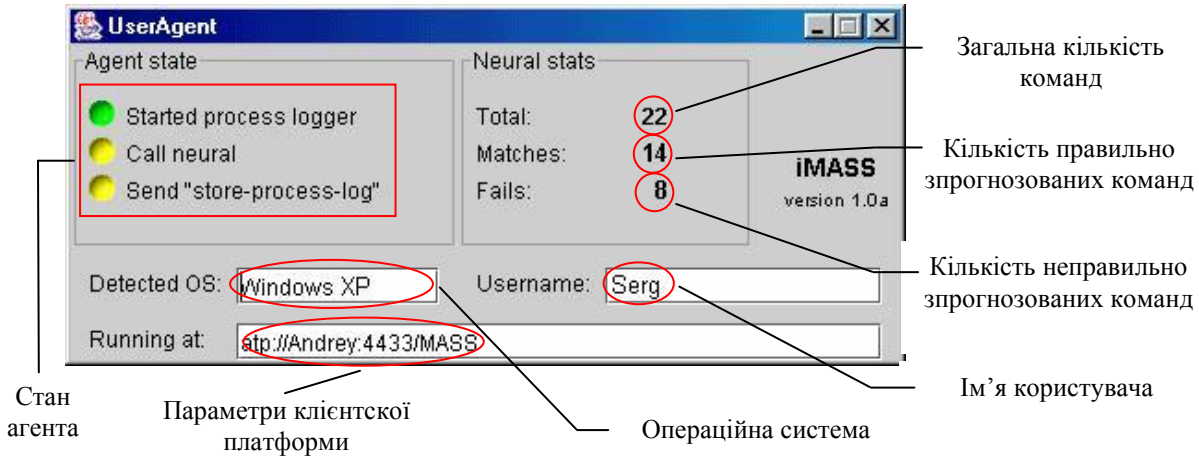


Рисунок 3.6 – Графічний інтерфейс користувача

У загальному випадку, задача оцінки ефективності роботи багатоагентної системи моніторингу діяльності користувачів комп'ютерних систем є достатньо складним і вимагає ретельного дослідження. У даній роботі буде застосований наступний простий підхід. Відомо, що близько 80-90% атак ініціюється користувачами всередині комп'ютерної системи. Пропонована багатоагентна система моніторингу дозволяє в середньому в 80% випадків виявляти аномальну поведінку користувачів. Припустимо, що 10-20% типів атак, що залишилися успішно виявляються іншими засобами. Тоді ймовірність виявлення атаки складає приблизно 0,8. Якщо вважати, що існуючі методи і підходи забезпечення безпеки

(такі як системи виявлення і запобігання вторгненням, антивірусні програми, різноманітні засоби адміністратора і інше) дозволяють виявити і запобігти в середньому 50% атакам, то використання багатоагентної системи аналізу поведінки користувачів в комп'ютерних мережах дозволяє збільшити її захищеність від внутрішніх загроз в середньому на 30%.

Проте відзначимо, що статистика ефективності засобів виявлення вторгнень і аномалій відсутня, тому оцінки є приблизними, а саме питання оцінки ефективності засобів інформаційної безпеки вимагає окремого опрацьовування.

У даному розділі розглянуті питання, пов'язані з програмною реалізацією комплексної моделі користувачів комп'ютерних систем. Для реалізації системи був використаний агентний підхід. Це дало можливість використовувати запропоновану систему моніторингу в гетерогенному середовищі (з різними операційними системами і форматами даних), а також забезпечити автономність і мобільність компонентів (можливість переміщення моделі на сторону користувача). Використання агентної технології дозволило зробити систему розподіленою і легко масштабованою. Так, при додаванні в систему нового користувача необхідно тільки внести до бази даних відповідну інформацію про модель користувача, а все решта система виконає сама.

В якості технології реалізації і середовища програмування мобільних агентів були вибрані, відповідно, Java і Aglets Software Development Kit (ASDK). Реалізована нейромережева система моніторингу підтвердила дієздатність і ефективність запропонованої моделі користувача.

ВИСНОВКИ

Дипломна робота присвячена розробці ефективних моделей аналізу поведінки користувачів комп'ютерних систем з метою їх застосування в системах моніторингу і виявлення аномальної діяльності. У роботі запропонована комплексна модель поведінки користувачів, побудована генеративна модель і реалізована комплексна модель у вигляді багатоагентної системи виявлення аномалій в поведінці користувачів комп'ютерних систем.

1. Аналіз існуючих методів побудови моделей поведінки користувачів комп'ютерних систем показав, що існуючі підходи не описують всі аспекти поведінки користувачів комп'ютерних систем. Так, в більшості робіт враховують тільки динамічні властивості поведінки користувачів, або тільки статистичну інформацію. При цьому ні як не враховують можливі тренди зміни поведінки користувачів.

2. Розроблена комплексна нейромережева модель поведінки користувачів комп'ютерних систем, що включає три компоненти: інтерактивну (прогнозну) складову, що враховує динаміку поведінки користувачів, сеансову (статистичну) складову, що враховує статистичні властивості поведінки користувача і модуль аналізу трендів, призначений для виявлення можливих зміни в поведінці користувачів.

3. На відміну від існуючих підходів, запропонована модель дозволяє забезпечити комплексний підхід до аналізу поведінки користувачів як під час його роботи (у режимі реального часу), так і після закінчення сеансу (у відкладеному режимі).

4. Розроблена генеративна модель поведінки користувачів, що забезпечує репрезентативні набори даних для ідентифікації і верифікації комплексної моделі. Для її побудови виконувалося статистичне моделювання даних. В якості критерію відповідності гіпотези реальним даним використовувався критерій χ^2. На основі отриманих розподілів згенерований необхідний набір даних, що дало можливість оптимізувати архітектуру нейронної мережі для сеансової моделі і покращити її функціонування.

5. Розроблена об'єктно-орієнтована модель представлення нейромережевих моделей користувача в рамках агентної парадигми. Запропоновані принципи функціонування і взаємодії агентів, що реалізовують нейромережеві моделі поведінки користувачів. Агентна реалізація дозволила використовувати розроблену систему в гетерогенному середовищі, забезпечити автономність і мобільність компонентів, дозволила зробити систему розподіленою і легко масштабованою. Використання багатоагентної системи аналізу поведінки користувачів в комп'ютерних мережах дозволяє збільшити її захищеність в середньому на 30%.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Назаров А.А. Морфологическое прогнозирование развития военной техники. – МО СССР, 1986. – 252 с.
2. Прогностика. Термины и определения / Комитет научно-технической терминологии, выпуск 109. – М.: Наука, 1990. – 56 с.
3. Касаев О.Б., Савченко В.И. Модели и методы прогнозирования технического состояния космических средств: Метод. Пособие. – СПб.: ВИКУ им. А.Ф. Можайского, 1997. – 37 с.
4. Гаскаров Д.В., Голинкевич Т.А., Мозгалевский А.В. Прогнозирование технического состояния и надежности радиоэлектронной аппаратуры. – М.: Сов. Радио, 1974. – 224 с.
5. Блинов И.Н., Гаскаров Д.В., Мозгалевский А.В. Автоматический контроль систем управления. – Л.: Энергия, 1968. – 165 с.
6. Смирнов Н.В., Дунин-Барковский И.В. Курс теории вероятностей и математической статистики. – М.: Наука, 1969. – 235 с.
7. Ивахненко А.Г. Самообучающиеся системы распознавания и автоматического управления. – К.: Техника, 1969. – 245 с.
8. Растригин Л.А., Пономарев Ю.П. Экстраполяционные методы проектирования и управления. – М: Машиностроение, 1986. – 120 с.
9. Горелик А.Л., Скрипкин В.А. Методы распознавания: Учеб. пособие. – М.: Высш. шк., 1984. – 208 с.
10. Haykin S. Neural Networks: a comprehensive foundation. – Upper Saddle River, New Jersey: Prentice Hall, 1999. – 842 p.
11. Cover T.M., Hart P.E. Nearest neighbor pattern classification // IEEE Transactions on Information Theory. – 1967. – vol. IT-13. – P. 21–27.
12. Нильсон Н. Обучающиеся машины. Пер. с англ. – М.: Мир, 1967. – 341 с.
13. Greenberg S. The Computer User as Toolsmith: The Use, Reuse, and Organization of Computer-based Tools. – New York, NY: Cambridge University Press, 1993.
14. Greenberg S., Witten I.H. How Users Repeat Their Actions on Computers: Principles for Design of History Mechanisms // In Proc. of CHI'88. – 1988. – P. 171-

178.

15. Tauscher L., Greenberg S. How people revisit Web pages: Empirical findings and implication for the development of history systems // International J. of Human Computer Studies. – 1997. – 47(1). – P. 97-138.
16. Debevc M., Meyer B, Svecko R. An adaptive short list for documents on the world wide web // Proc. of the 1997 International Conf. on Intelligent User Interfaces. – Orlando, FL (USA). – 1997. – P. 209-211.
17. Masui T., Nakayama K. Repeat and Predict – Two Keys to Efficient Text Editing // In Proc. of CHI'94. – 1994. – P. 118-123.
18. Manavoglu E., Pavlov D., Lee Giles C. Probabilistic User Behavior Models // Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003). – Melbourne, Florida (USA). – 2003. – P. 203–210.
19. Davison B. D., Hirsh H. Probabilistic Online Action Prediction // Working Notes of the AAAI Spring Symposium on Intelligent Environments. – 1998. – P. 148–154.
20. Куссуль Н., Соколов А. Адаптивное обнаружение аномалий в поведении пользователей компьютерных систем с помощью марковских цепей переменного порядка. Часть 1. Адаптивная модель марковских цепей переменного порядка // Проблемы управления и информатики. – 2003. – № 3. – С. 83-93.
21. Куссуль Н.Н., Соколов А.М. Адаптивное обнаружение аномалий в поведении пользователей компьютерных систем с помощью марковских цепей переменного порядка. Часть 2. Методы обнаружения аномалий и результаты экспериментов // Проблемы управления и информатики. – 2003. – № 4. – С. 83–88.
22. Ryan J., Lin M-J., Miikkulainen R. Intrusion Detection with Neural Networks // Advances in Neural Information Processing Systems. – Cambridge, MA: MIT Press, 1998. – 10. – P. 943–949.
23. Безмалый В. Программные средства для наблюдения за персоналом // Компьютеры+Программы. – 2002. – № 2. – С. 42–46.
24. Debar H., Dacier M., Wespi A. Towards a taxonomy of intrusion-detection systems // Computer Networks. – 1999. – Vol. 31, Issue 8. – P. 805–822.

25. Garvey T., Lunt T. Model-based intrusion detection // Proc. 14th National Computer Security Conf. – 1991. – P. 372–385.
26. Kumar S., Spafford E. A pattern matching model for misuse intrusion detection // Proc. 17th National Computer Security Conf. – 1994. – P. 11–21.
27. Helman P., Liepins G. Statistical foundations of audit trail analysis for the detection of computer misuse // IEEE Transactions on Software Engineering. – 1993. – 19 (9). – P. 886–901.
28. Dowell C., Ramstedt P. The ComputerWatch data reduction tool // Proc. 13th National Computer Security Conf. – Washington, DC (USA). – 1990. – P. 99–108.
29. Ryan J., Lin M-J., Miikkulainen R. Intrusion Detection with Neural Networks // Advances in Neural Information Processing Systems. – Cambridge, MA: MIT Press, 1998. – 10. – P. 943–949.
30. Резник А.М., Куссуль Н.Н., Соколов А.М. Нейросетевая идентификация поведения пользователей компьютерных систем // Кибернетика и вычислительная техника. – 1999. – № 123. – С. 70-79.
31. PC Spy – http://www.softdd.com/pcspy/index.htm.
32. Inlook Express – http://www.jungle-monkey.com.
33. Paparazzi – http://www.industar.net.
34. Hirsh H., Davison B.D. An Adaptive unix Command-Line Assistant // In Proc. of the First International Conference on Autonomous Agents. – 1997. – P. 542-543.
35. Yoshida K., Motoda H. Automated user modeling for intelligent interface // International J. of Human-Computer Interaction. – 1996. – 8(3). – P. 237-258.
36. Motoda H., Yoshida K. Machine Learning Techniques to Make Computers Easier to Use // In Proc. of the 15th International Joint Conf. on Artificial Intelligence, Morgan Kaufmann. – 1997. – P. 1622-1631.
37. Lee A. Investigations into History Tools for User Support: Ph.D. Dissertation. – University of Toronto, 1992.
38. Joachims T., Freitag D., Mitchel T. Web-watcher: A tour guide for world wide web // Proceedings of the Fifteenth International Joint Conference on Artificial intelligence. – 1997. – P. 770–775.
39. Spiliopoulou M., Faulstich L., Winkler K. A Data Miner Analyzing the Navigational

Behaviour of Web Users // In Proc. of the Workshop on Machine Learning in User Modeling: Advanced Course on Artificial Intelligence (ACAI'99). – Chania (Greece). – 1999.

40. Perkowitz M., Etzioni O. Adaptive Web Sites: Conceptual Cluster Mining // Sixteenth International Joint Conf. in Artificial Intelligence. – Stockholm, Sweden. – 1999. – P. 264-269.
41. Ночевнов Д.П. Метод адаптивного информационного поиска на основе контекстной модели пользователя // Вісник Черкаського державного технологічного університету. – 2003. – № 3. – С. 17-23.
42. Moukas A. Amalthaea: Information Discovery and Filtering using a Multiagent Evolving Ecosystem // Applied Artificial Intelligence: An International Journal. – 1997. – 11(5). – P. 437-457.
43. Virvou M., Moundridou M. Student and Instructor Models: Two Kinds of User Model and their Interaction in an ITS Authoring Tool / In: Bauer M., Gmytrasiewicz P., Vassileva J. (eds.): User Modeling 2001 // Proc. of the 8th International Conference UM2001, Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science). – Vol. 2109. – Springer, Berlin. – 2001. – P. 158-167.
44. Chiu B.C., Webb G. Dual-model: An Architecture for Utilizing Temporal Information in Student Modeling // In Proc. of the Workshop on Machine Learning in User Modeling: Advanced Course on Artificial Intelligence (ACAI'99). – Chania (Greece). – 1999.
45. Dorronsoro J.R., Ginel F., S´anchez C., Cruz C.S. Neural fraud detection in credit card operations // IEEE Transactions on Neural Networks. – 1997. – 8(4). – P. 827–834.
46. Leonard K.J. Detecting credit card fraud using expert systems // Computers and Industrial Engineering. – 1993. – 25(1-4). – P. 103–106.
47. Ghosh S., Reilly D.L. Credit card fraud detection with a neural network // In Proc. of the Twenty-Seventh Hawaii International Conference on System Sciences, IEEE Computer Society Press. – 1994. – P. 621–630.
48. Hanagandi V., Dhar A., Buescher K. Density-based clustering and radial basis

function modeling to generate credit card fraud scores // In Proc. of the IEEE/IAFE 1996 Conference on Computational Intelligence for Financial Engineering (CIFEr), IEEE Press. – 1996. – P. 247–251.

49. Aleskerov E., Freisleben B., Rao B. CARDWATCH: A neural network based database mining system for credit card fraud detection // In Proc. of the IEEE/IAFE 1996 Conference on Computational Intelligence for Financial Engineering (CIFEr), IEEE Press. – 1996. – P. 220–226.
50. Stolfo S.J., Fan D.W., Lee W., Prodromidis A.L. Credit card fraud detection using meta-learning: Issues and initial results // In Proc. Of AAAI-97 Workshop on AI Approaches to Fraud Detection & Risk Management, AAAI Press. – 1997. – P. 83–90.
51. Вентцель Е.С. Теория вероятностей: Учебник для студ. вузов. – М.: Издательский центр «Академия», 2003. – 576 с.
52. Russel, S., Norvig, P. Artificial Intelligence: A Modern Approach. – Upper Saddle River NJ: Prentice Hall, 1995. – 932 p.
53. Luck M., McBurney P., Preist C. Agent Technology: Enabling Next Generation Computing. – N.Y.: AgentLink, 2003. – 94 p.
54. Wooldridge M., Jennings N.R. Intelligent agents: Theory and practice // The Knowledge Engineering Review. – 1995. – Vol. 10, № 2. – P. 115–152.
55. Jennings N.R., Sycara K., Wooldridge M. A Roadmap of Agent Research and Development // Autonomous Agents and Multi-Agent Systems. – 1998. – Vol. 1. – P. 275–306.
56. Aglets Software Development Kit. – http://sourceforge.net/projects/aglets/.
57. Oshima M., Karjoth G. Aglets Specification (1.0) – http://www.trl.ibm.com/aglets/spec10.htm.
58. Методичні рекомендації до виконання дипломної роботи з освітньо-кваліфікаційного рівня “Магістр”. Спеціальність „Комп’ютерні системи та мережі” / О.М. Березький, Р.Б. Трембач, Г.М. Мельник / Під ред. О.М. Березького – Тернопіль: ТНЕУ, 2012.– 42 с.

Додаток А

Програмний код реалізації нейромережевих моделей

Клас `Layer`, який реалізовує шар нейронної мережі.

```
package neural;
import java.io.Serializable;
public class Layer implements Serializable
{
        Math m;
        boolean bLast = false;
        double dMaxNum;
        int LSize;
        int prevLSize;
        int MtxSize;
        double[] Bias;
        double[] Weights;
        double[] Outputs;

        public Layer (int CurSize, int PrevSize)
        {
                LSize     = CurSize;
                prevLSize = PrevSize;
                MtxSize   = LSize*prevLSize;
                Bias      = new double[LSize];
                Weights   = new double[MtxSize];
                Outputs   = new double[LSize];
        }

        void Run (double[] prevOut)
        {
                int Index = 0;
                double dTmp;
                for (int i=0; i<LSize; i++)
                {
                        dTmp = Bias[i];
                        for (int j=0; j<prevLSize; j++)
                        {
                          dTmp += prevOut[j]*Weights[Index];
                          ++Index;
                        }
                        if (bLast) Outputs[i] =
dLastFunction(dTmp);
                        else Outputs[i] = dTrFuntion(dTmp);
                }
        }

        double dTrFuntion (double X)
        {
          if (40.0<X)  return 1.0;
                if (X<-40.0) return 0.0;
                return 1.0/(1.0 + m.exp(-X));
        }
```

```
        double dLastFunction (double X)
        {
          if (dMaxNum/2<X)  return dMaxNum;
                if (X<-dMaxNum/2) return 0.0;
                return X+dMaxNum/2;
        }
}
```

Клас `TestNeural`, який реалізовує режим прогону нейронної мережі.

```
package neural;

import java.io.*;
import java.util.Properties;
import java.io.Serializable;

public class TestNeural implements Serializable
{
        Layer[] Layers;
        int LNum;
        double pdWeights[];
        public boolean CanRun = false;
        String NetFileName;
        String ParFileName;

public TestNeural (String Path) throws IOException
        {
                ParFileName = Path + ".par";
                NetFileName = Path + ".net";
         System.out.println("PATH1: " + ParFileName);
         System.out.println("PATH2: " + NetFileName);
                ReadArch();
                CanRun = ReadNet();
                SetWeights();
        }

        void ReadArch () throws IOException
        {
                String myString;
                Properties props = System.getProperties();
    InputStream is;
                        int Arch[] = new int[4];
                        is = new FileInputStream(ParFileName);
                        props.load(is);
                        LNum    =
(Integer.valueOf((String)props.get("train.parameters.0"))).intValu
e();
                        Layers  = new Layer[LNum];
                        Arch[0] =
(Integer.valueOf((String)props.get("train.parameters.1"))).intValu
e();
                        Arch[1] =
(Integer.valueOf((String)props.get("train.parameters.2"))).intValu
```

```
e();
                        Arch[2] =
(Integer.valueOf((String)props.get("train.parameters.3"))).intValu
e();
                        Arch[3] =
(Integer.valueOf((String)props.get("train.parameters.4"))).intValu
e();
                        int iNNum = 0;
                        switch (LNum)
                        {
                                case 3: Layers[2] = new Layer
(Arch[iNNum+1],Arch[iNNum]);

++iNNum;
                                case 2: Layers[1] = new Layer
(Arch[iNNum+1],Arch[iNNum]);

++iNNum;
                                case 1: Layers[0] = new Layer
(Arch[iNNum+1],Arch[iNNum]);

Layers[0].bLast = true;

Layers[0].dMaxNum  =
(Double.valueOf((String)props.get("train.parameters.12"))).doubleV
alue();

break;
                        }
                 iNNum = 0;
                        for (int i=0; i<LNum; i++) iNNum +=
Layers[i].MtxSize + Layers[i].LSize;
                        pdWeights = new double[iNNum];
        }

        boolean ReadNet () throws IOException
        {
                double[] WTmp;
                File FNet = new File (NetFileName);
                if (FNet.exists())
                {
                        try
                        {
                                ObjectInputStream OS = new
ObjectInputStream(new FileInputStream(FNet));
                                WTmp = (double[])OS.readObject();
                                if (WTmp.length !=
pdWeights.length) return false;
                                for (int i=0; i<WTmp.length; i++)
pdWeights[i] = WTmp[i];
                                OS.close();
                        }
                        catch (ClassNotFoundException e) { return
false; }
                        catch (EOFException e) { return false; }
```

```
                        return true;
                }
                else return false;
        }

        void SetWeights ()
        {
                int Seek = 0;
                for (int j=LNum-1; 0<=j; j--)
                {
                        for (int i=0; i<Layers[j].LSize; i++)
Layers[j].Bias[i] = pdWeights[Seek+i];
                        Seek += Layers[j].LSize;
                        for (int i=0; i<Layers[j].MtxSize; i++)
Layers[j].Weights[i] = pdWeights[Seek+i];
                        Seek += Layers[j].MtxSize;
                }
        }

        public int Test (double[] dpInp)
        {
                Layers[LNum-1].Run (dpInp);
                for (int i=LNum-1; 0<i; i--) Layers[i-
1].Run(Layers[i].Outputs);
                return (int)(Layers[0].Outputs[0] + 0.5);
        }
}
```

Клас `TrainNeural`, в якому реалізовані методи, які призначені для навчання нейронної мережі.

```
package neural;

import java.io.*;
import java.util.Properties;

public class TrainNeural extends TestNeural
{
        Math m;
        int iRecordLength  = 0;
        int iRecordsNumber = 0;
        double pdInputBuffer[];
        double pdInput[];
        int piClass[];
        int StopFlag = 5500000;

        long   lCheckPoint;
        int    lNumOfPoints;
        double dTrFactor;
        double dOffset;

        double LC[] = new double[3];
        double MC[] = new double[3];
```

```
        double MBias[][]   = new double[3][];
        double Moments[][] = new double[3][];
        double Errors[][]  = new double[3][];


        public TrainNeural (String Path) throws IOException
        {
                super(Path);
                ReadParameters();
                ReadData (Path + ".trn");
        }

        void ReadParameters ()
        {
                String myString;
                Properties props = System.getProperties();
    InputStream is;
                try
                {
                        is = new FileInputStream(ParFileName);
                        props.load(is);
                        switch (LNum)
                        {
                                case 3: LC[1]=
(Double.valueOf((String)props.get("train.parameters.6"))).doubleVa
lue();
                                                                MC[1]=
(Double.valueOf((String)props.get("train.parameters.9"))).doubleVa
lue();

LC[2]=
(Double.valueOf((String)props.get("train.parameters.5"))).doubleVa
lue();
                                                                MC[2]=
(Double.valueOf((String)props.get("train.parameters.8"))).doubleVa
lue();

break;
                                case 2: LC[1]=
(Double.valueOf((String)props.get("train.parameters.5"))).doubleVa
lue();
                                                                MC[1]=
(Double.valueOf((String)props.get("train.parameters.8"))).doubleVa
lue();
                        }
                        LC[0]=
(Double.valueOf((String)props.get("train.parameters.7"))).doubleVa
lue();
                        MC[0]=
(Double.valueOf((String)props.get("train.parameters.10"))).doubleV
alue();
                        lCheckPoint =
(Long.valueOf((String)props.get("train.parameters.14"))).longValue
();
                        dTrFactor   =
```

```
(Double.valueOf((String)props.get("train.parameters.13"))).doubleV
alue();
                        dOffset                 =
(Double.valueOf((String)props.get("train.parameters.16"))).doubleV
alue();
                        lNumOfPoints=
(Integer.valueOf((String)props.get("train.parameters.15"))).intVal
ue();
             StopFlag =
(Integer.valueOf((String)props.get("train.parameters.17"))).intVal
ue();

Rand((Double.valueOf((String)props.get("train.parameters.11"))).do
ubleValue());
                        SetWeights();
                        for (int i=0; i<LNum; i++)
                        {
                                MBias[i]   = new
double[Layers[i].LSize];
                    Moments[i] = new double[Layers[i].MtxSize];
                    Errors[i]  = new double[Layers[i].LSize];
                        }
                }
                catch(Exception ex)
                {
      ex.printStackTrace();
                }
        }

        void Rand (double Range)
        {
                for (int i=0; i<pdWeights.length; i++)
pdWeights[i] = Range*(2.0*m.random() - 1.0);
        }

        void GetWeights ()
        {
                int Seek = 0;
                for (int j=LNum-1; 0<=j; j--)
                {
                        for (int i=0; i<Layers[j].LSize; i++)
pdWeights[Seek+i] = Layers[j].Bias[i];
                        Seek += Layers[j].LSize;
                        for (int i=0; i<Layers[j].MtxSize; i++)
pdWeights[Seek+i] = Layers[j].Weights[i];
                        Seek += Layers[j].MtxSize;
                }
        }

        boolean ReadData (String datFile) throws IOException
        {
                String myString;
                int i, iIndex = 0;
                File FDat = new File (datFile);
                double[] buf;
```

```
                iRecordsNumber = 0;
                if (!FDat.exists()) return false;

                BufferedReader in = new BufferedReader(new
InputStreamReader

(new DataInputStream(new FileInputStream(FDat))));
                try
                {
                        myString = in.readLine();
                        iRecordLength = CheckString(myString);
                        --iRecordLength;
                        iRecordsNumber = 1;
                        while (true)
                        {
                                myString = in.readLine();
                                if (myString == null)   break;
                                ++iRecordsNumber;
                        }
                }
                catch (EOFException e){ }
                catch (IOException e) { iRecordsNumber = 0; }
                in.close();
                buf = new double[iRecordLength+1];
                pdInput = new double[iRecordLength];
                if ((0<iRecordLength)&&(0<iRecordsNumber))
                {
                        piClass = new int[iRecordsNumber];
                        pdInputBuffer = new
double[iRecordLength*iRecordsNumber];
                        BufferedReader in1 = new
BufferedReader(new InputStreamReader

(new DataInputStream(new FileInputStream(FDat))));
                        try
                        {
                                for (int j=0; j<iRecordsNumber;
j++)
                                {
                                        myString = in1.readLine();
                                        if (myString == null)
break;
                                        if ((iRecordLength+1) !=
ReadString (myString,buf)) break;
                                        for (i=0; i<iRecordLength;
i++) pdInputBuffer[iRecordLength*j+i] = buf[i];
                                        piClass[j] =
(int)buf[iRecordLength];
                                }
                        }
                        catch (EOFException e){ }
                        catch (IOException e) { iRecordsNumber =
0; }
                        in1.close();
                }
```

```
                return 0<iRecordsNumber;
        }

        int CheckString (String s)
        {
                int spaceAt, startingFrom = 0, count = 0;
                while (true)
                {
                        spaceAt = s.indexOf(" ",0);
                        if (spaceAt == -1) break;
                        if (spaceAt == 0) s =
s.substring(1,s.length());
                        else
                        {
                                ++count;
                                s =
s.substring(spaceAt+1,s.length());
                        }
                }
                if (0<s.length())       ++count;
                return count;
        }

        int ReadString (String s, double[] buf)
        {
                int spaceAt, startingFrom, count = 0;
                Double dA;
                startingFrom = 0;
                while (true)
                {
                        spaceAt = s.indexOf(" ",0);
                        if (spaceAt == -1) break;
                        if (spaceAt == 0) s =
s.substring(1,s.length());
                        else
                        {
                                dA =
Double.valueOf(s.substring(0,spaceAt));
                                buf[count] = dA.doubleValue();
                                ++count;
                                s =
s.substring(spaceAt+1,s.length());
                        }
                }
                if (0<s.length())
                {
                        dA = Double.valueOf(s);
                        buf[count] = dA.doubleValue();
                        ++count;
                }
                return count;
        }

        void ErrorCalc ()
        {
```

```
                int Index;
                for (int k=0; k<LNum-1; k++)
                {
                        for (int j=0; j<Layers[k+1].LSize; j++)
                        {
                                Index = j;
                                Errors[k+1][j] = 0;
                                for (int i=0; i<Layers[k].LSize;
i++)
                                {
                                        Errors[k+1][j] +=
Errors[k][i]*Layers[k+1].Weights[Index];
                                        Index +=
Layers[k+1].LSize;
                                }
                                Errors[k+1][j] *= (dOffset +
Layers[k+1].Outputs[j]*(1-Layers[k+1].Outputs[j]));
                        }
                }
        }

        void Update (int k, double[] Inp)
        {
                int Index = 0;
                for (int i=0; i<Layers[k].LSize; i++)
                {
                        MBias[k][i] = MC[k]*MBias[k][i] +
LC[k]*Errors[k][i];
                        Layers[k].Bias[i] += MBias[k][i];
                        for (int j=0; j<Layers[k].prevLSize; j++)
                        {
                                Moments[k][Index] =
MC[k]*Moments[k][Index] + LC[k]*Errors[k][i]*Inp[j];
                                Layers[k].Weights[Index] +=
Moments[k][Index];
                                ++Index;
                        }
                }
        }
        boolean ErrorPropagation (int iRecPointer)
        {
                int i, j;
                double dTmp;

                for (j=0; j<iRecordLength; j++)
                {
                        pdInput[j] =
pdInputBuffer[iRecPointer*iRecordLength+j];
                }
                int iTmpClass;
                iTmpClass = Test(pdInput);
          if (piClass[iRecPointer] == iTmpClass) return false;
                Errors[0][0] = (piClass[iRecPointer] -
Layers[0].Outputs[0])/Layers[0].dMaxNum;
```

```
                ErrorCalc();
                Update(LNum-1,pdInput);
                for (i=LNum-1; 0<i; i--) Update(i-
1,Layers[i].Outputs);
                return true;
        }

        boolean WriteNet() throws IOException
        {
                GetWeights();
                File FNet = new File (NetFileName);
                ObjectOutputStream OS = new ObjectOutputStream(new
FileOutputStream(FNet));
          OS.writeObject(pdWeights);
          OS.close();
          return true;
        }

        public int Train (int FrameRate) throws IOException
        {
                int j, i = 0, iCount = 0, iErrCount = 0;
                while ((i<StopFlag)||(iErrCount==0.0))
                {
                  if (ErrorPropagation(iCount)) ++iErrCount;
                        ++iCount;
                        iCount = (iCount<iRecordsNumber) ? iCount
: 0;

                        if (i==lCheckPoint)
                        {
                                for (j=0; j<LNum; j++)
                                {
                                        LC[j] *= dTrFactor;
                                        MC[j] *= dTrFactor;
                                }
                                lCheckPoint =
(long)(lCheckPoint/dTrFactor);
                        }

                        if (i%FrameRate == (FrameRate-1))
                        {

System.out.print((100.0*(double)iErrCount)/FrameRate + " ");
                                iErrCount = 0;
                        }
                        ++i;
                }
                WriteNet();
                return 0;
        }
}
```

Додаток Б

Довідка про використання