УДК 517.9

ЗАСТОСУВАННЯ НАВЧАННЯ З ПІДКРІПЛЕННЯМ ДЛЯ ПОБУДОВИ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

Б. Романюк, О. Пелюшкевич, Ю. Щербина

Львівський національний університет імені Івана Франка,
вул. Університетська, 1, Львів, 79000,
e-mail: bohdan2307@gmail.com, olga.peliushkevych@lnu.edu.ua,
yuriy.shcherbyna@lnu.edu.ua

Розглянуто застосування машинного навчання з підкріпленням для побудови рекомендації на основі позитивної кооперації користувачів із системою. Такий підхід, на відміну від інших, є динамічним, тобто адаптується до змін вподобань користувача та дає змогу зосередитися на максимізації вигоди від взаємодії користувача з рекомендаційною системою. Продемонстровано приклад використання такого підходу для рекомендації фільмів на основі оцінок користувачів.

Ключові слова: машинне навчання з підкріпленням, нейронна мережа, подання процесу прийняття рішень Маркова, рекомендаційна система, Q-навчання, Policy-навчання.

1. Вступ

В епоху суттєвого зростання кількості інформації у всесвітній мережі Інтернет постає потреба її рекомендації для покращення доступності, швидкості аналізу та сприйняття. Актуальність тематики полягає в тому, що генерування рекомендаційних вибірок важлива у багатьох сферах людської діяльності, починаючи від індустріальних і наукових потреб, закінчуючи бізнес потребами. Побудова рекомендацій особливо актуальна в сучасному світі, адже кількість інформації у всесвітній павутині зростає нечуваними темпами та постає проблема для користувачів, яким потрібно знайти певну інформацію, яка б була для них цікава, та для бізнесу, аби зацікавити користувача і він віддав перевагу саме їхній системі, а не конкурентам. На жаль, більшість сучасних ресурсів і сервісів використовують для цього статичні підходи та методи засновані на фільтрації змісту [11], матричній факторизації [6], наївному класифікаторі Баєса [1, 12], логістичній регресії [1], методах з використанням багаторукого бандита [7, 18]. Сьогодні ці статичні методи та підходи використовують такі великі компанії-гіганти: Amazon [9], Netlix [4, 15] та Google [2]. Проте вони мають кілька недоліків, серед яких нерозуміння того, що взаємодія користувачів із рекомендаційною системою є послідовною, внаслідок чого орієнтуються лише на конкретну дискретну подію в певний момент часу. Інший важливий недолік статичного підходу – не орієнтованість на довготривалій винагороді від побудованої результуючої пропозиції. Саме тому перспективно використовувати нейронні мережі та машинне навчання з підкріпленням [16], які дають змогу уникнути недоліків статичних підходів для розв'язання таких задач. Перевага машинного навчання з підкріпленням полягає в тому, що воно легко може працювати в середовищі, де є послідовна взаємодія користувачів із системою і направлене на розв'язання проблеми, що і як виконувати в цьому середовищі для максимізації такого узагальненого поняття як винагорода. Крім того, цей підхід зосереджений на довготривалій взаємодії користувачів із системою. Тобто він

ідеальний у системах, де потрібно побудувати рекомендаційний набір елементів на підставі взаємодії користувачів. Варто зауважити, що ця модель навчання дуже схожа на те, як навчаються люди, тобто намагається досягти ідеалу в процесі навчання.

Ми розглянемо та продемонструємо підхід із використанням машинного навчання з підкріпленням для процесу послідовної побудови рекомендацій у вигляді моделі "Актор-Критик" [19], яка дає змогу генерувати рекомендаційну вибірку динамічно, зосереджується на довготривалій взаємодії користувачів із системою. Під динамічною побудовою мається на увазі те, що система зможе динамічно адаптуватися до зміни вподобань користувачів. Довготривала винагорода відповідає за побудову такого результуючого набору, який задовольняв би потреби кожного окремого користувача в якомога більшій мірі. Першу перевагу легко пояснити на такому прикладі, скажімо користувач деякої системи переглядав відео на історичну тематику протягом певного періоду, а потім у певний момент почав переглядати науково-фантастичні відео, статична система в цьому випадку одразу почне рекомендувати тільки науково-фантастичні відео, натомість динамічний підхід буде і надалі генерувати результуючу вибірку на підставі історичних відео, поступово додаючи до цієї вибірки науково-фантастичні відео. Другу перевагу також легко пояснити на цьому прикладі, скажімо система запропонувала користувачеві певне відео, і користувач ним зацікавився, переглянув та оцінив, тобто така взаємодія була позитивною для користувача та для самої системи, натомість якщо користувачеві не сподобається один елемент з пропозиції, то він може оцінити його низькою оцінкою, внаслідок чого система не отримає виграшу від цього і зрозуміє, що певні типи відео не варто пропонувати користувачеві, адже від цього немає виграшу ні для системи, ні для користувача. В статті буде продемонстровано рекомендацію фільмів на підставі оцінок користувачів, хоча загалом такий підхід можна буде використати для будь-якого набору даних: новини, техніка, товари, послуги і т.д.

2. Формулювання задачі

Задача полягає у тому, що потрібно розробити:
- систему для рекомендації фільмів з використанням машинного навчання з підкріпленням, моделі "Актор-Критик" та градієнта глибокої детермінованої стратегії;
- механізм для збору та заповнення даних;
- модуль подання даних та активностей користувачів у певному вигляді, який може бути використаний інтерфейсом нейронної мережі;
- механізм навчання та тестування на підставі цих даних;
- порівняльну характеристику навчальної, тестової вибірок і точність системи.

3. Машинне навчання з підкріпленням

Машинне навчання з підкріпленням [16] – це сфера машинного навчання, яка зосереджена на вирішенні питання про те, які дії мають виконувати програмні агенти в певному заданому середовищі для максимізації такого абстрактного поняття – винагорода. До переваг цього принципу навчання можна зарахувати орієнтованість на великі обсяги даних, довготривалу побудову результуючого набору та адаптивність до змін у середовищі. Навчання з підкріпленням є однією з основних парадигм машинного навчання і широко використовується у робототехніці, роботизованих системах, фінансових системах та для побудови рекомендацій.

Рис. 1. Взаємодія користувачів із рекомендаційною системою

На рис. 1 проілюстровано взаємодію між користувачами та програмним агентом в деякому середовищі. Часто такий процес називають Марковським процесом прийняття рішення [16, 19], який є кортежем з п'яти елементів (S, A, P, R, γ). Цей процес і є основою машинного навчання з підкріпленням.

Подання процесу прийняття рішень Маркова

- **Простір стану** S. Стан $s_t = s_t^1, ..., s_t^N \in S$ – це історія взаємодії користувача з системою, тобто останні N елементів оцінених користувачем до моменту t. Елементи в s_t посортовані в хронологічному порядку.
- **Простір дій** A. Дія $a_t = a_t^1, ..., a_t^K \in A$ – це рекомендований список елементів для користувача в момент t на підставі поточного стану s_t, де K - це число елементів, які рекомендуються користувачеві в кожен момент часу.
- **Винагорода** R. Після побудови рекомендації на підставі дії a_t та стану s_t, рекомендована вибірка надається користувачеві для перегляду, той може її оцінити(пропустити) і "Агент" отримає винагороду $r_t = R(s_t, a_t)$ відповідно до відгуку користувача.
- **Ймовірність переходу** P. Ймовірність переходу $p(s_{t+1}|s_t, a_t)$ визначає ймовірність переходу зі стану s_t в стан s_{t+1}, коли "Агент" отримує дію a_t. Якщо користувач проігнорував всі рекомендовані елементи, тоді наступний стан $s_{t+1} = s_t$, однак коли він оцінить деякі елементи з вибірки, то система перейде в стан s_{t+1}.
- **Коефіцієнт знижки** γ. $\gamma \in [0, 1]$ визначає коефіцієнт знижки, коли вимірюється вартість майбутньої винагороди. Коли $\gamma = 0$ система розглядає лише негайну винагороду, коли ж $\gamma = 1$ майбутні винагороди можна вирахувати на підставі поточних дій.

У деякому середовищі в певний момент t всі дії a_t користувача в системі винагороджуються певним числовим значенням r_t. Середовище надає агентові список усіх станів користувача s_t(позитивних взаємодій користувача з системою) в момент часу t і в цей самий момент часу обчислюється винагорода r_t. Агент генерує набір результуючих дій(рекомендацій) a_{t+1} і процес переходить до наступного моменту $t + 1$. Головне завдання такого процесу – максимізація винагороди r_t від так згенерованого набору дій(рекомендацій) a_{t+1}.

4. Збір і подання даних

Для збору даних про фільми було розроблено спеціальну програму парсер з використанням мови програмування *C#*, яка дала змогу зібрати інформацію про понад 10 тисяч фільмів з сайтів *https://www.imdb.com/*, *https://omdbapi.com* та *https://themoviedb.org*. Навчання відбуватиметься на наборі даних, який має назву *Movie Lens* (оцінки користувачів з сайту *https://movielens.org*) і містить близько 20 мільйонів оцінок користувачів. Варто зауважити, що цей набір даних було частково змінено, оскільки парсер для уніфікації кожного фільму використовував його номер з сайту *https://imdb.com*, а *Movie Lens* містить свій унікальний ідентифікатор, то було вирішено використовувати в обох випадках унікальний номер з сайту *https://imdb.com*. Позаяк було збережено лише 10 тисяч фільмів, то, відповідно, і кількість відгуків потрібно зменшити, адже в процесі навчання може виникнути ситуація, що певні оцінки не будуть мати відповідного їм фільму, бо його немає у базі даних. Для цього також було розроблено консольну програму з використанням мови програмування *C#*, яка підготувала оцінки користувачів для подальшого використання. У підсумку отримано близько 5,4 мільйони оцінок для навчання та тестування.

Зважаючи на те, що нейронні мережі не можуть працювати з абстракцією фільмів напряму, а лише з числовими та векторними зображеннями, то для подальшого аналізу, навчання і тестування потрібно отримати векторне подання всіх фільмів на підставі унікальних властивостей кожного з них, що дасть змогу з'ясувати, наскільки вони схожі між собою [5]. До властивостей, які однозначно ідентифікують певний фільм, можна зачислити такі: рік виходу, жанри, країни виробництва, мова, режисер, кінокомпанія, сюжет, рейтинги на різних сайтах, кількість голосів, популярність, тривалість, бюджет, касові збори та інші.

Рис. 2. Подання жанрів за допомогою бітових масок

З жанрами, країнами виробництва, рейтингами, кількістю голосів тощо все досить просто, адже вони повторюються і для них можна задати порядкові цілі числа, починаючи від одиниці або ж ці властивості самі є числовими компонентами (рейтинг, кількість голосів, тривалість у хвилинах і т.д.). Варто зауважити, що кожен фільм може містити декілька жанрів, країн виробництва чи розмовних мов, тому для подання цих даних використано масиви бітових масок. Тобто, наприклад, якщо всього є 104 жанри, то генерується масив із 104 елементів, заповнений нулями, а для жанрів, які відповідають цьому фільму у масиві, проставляється число 1 відповідно до порядкового номера жанру у цьому масиві. Наприклад, на рис. 2 показано приклад, коли жанри посортовані за алфавітом, і певний фільм належить до жанрів арт-хаус і біографічний. Число одиниця в такому випадку інтерпретується як істинна та означає, що цей фільм належить до цього жанру.

Аналогічно будуються бітові маски для подання країн і мов, якими розмовляють у фільмах. У підсумку отримано, що бітові масиви жанрів для кожного фільму мають розмірність 104 елементи, масиви країн – 114 елементів і масив розмовних мов – 176 елементів.

Текстові поля (сюжет), зручно подавати у вигляді векторів певної розмірності [14]. Перед початком роботи з текстовими елементами, треба видалити з них усі шумові слова, тобто слова, які не несуть жодного смислового навантаження (це числівники, частки та інше), видалити знаки пунктуації, а всі слова звести до одного регістру, наприклад, нижнього. Далі для перетворення тексту у масив використано підхід *word2vec* [3], який є набором технологій для подання слів у векторному вигляді, розроблений компанією Google у 2013 році. В такому випадку кожне слово є точкою у певному багатовимірному просторі, а основною перевагою такого підходу є те, що схожі за значенням слова у семантичному значенні мають меншу відстань між собою, ніж з іншими словами. Це легко уявити, якщо взяти за розмірність векторів число 3, тоді схожі за значенням слова будуть утворювати певні кластери у декартовій системі координат. За розмірність текстового поля сюжет було вирішено взято число 512, яке є достатнім числом для зберігання основних ознак. Цей етап надзвичайно важливий, адже сюжет єдина текстова властивість, яка буде подана для кожного елемента вибірки у вигляді одновимірних векторів заданого розміру, що складаються з чисел.

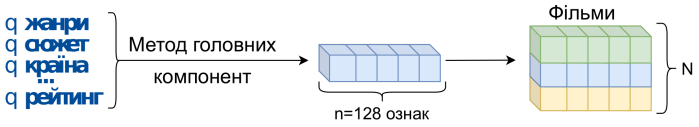

Рис. 3. Векторне подання фільмів

Далі всі згенеровані масиви ознак конкатенуються і для зручного об'єднання всіх характеристик застосовується метод головних компонент [13] для визначення важливості кожної окремої властивості та коригування результуючого вектора (див. рис. 3). За розмірність результуючого вектора, тобто вектора для подання кожного фільму було обрано розмірність $n = 128$, цієї кількості ознак достатньо для однозначної ідентифікації того чи іншого фільму. Так вдається зменшити розмірність об'єднаних початкових векторів до векторів розмірності 128, з яким буде простіше та швидше працювати система в подальшому.

5. Структура системи та методи навчання

Для навчання було обрано два методи: Policy-навчання та Q-навчання. Policy-навчання зосереджене на побудові подібних елементів, які б зацікавили користувача. Q-навчання тренує мережу для отримання якомога більшої винагороди, тобто збільшення зацікавленості користувачів у запропонованій рекомендації і покращенні наступних так побудованих вибірок. Ми розглянемо одночасне застосування цих двох підходів машинного навчання, щоб побудувати якомога ліпшу вибірку елементів. Зауважимо, що кожен з цих підходів має свої переваги та недоліки. Наприклад, Q-навчання на великій кількості даних стає дуже неефективним і потребує великої кількості часу для навчання. Натомість Policy-навчання збігається швидше

на великих об'ємах даних, але воно не розуміє послідовності дій користувачів на відміну від Q-навчання. З цих двох причин було вирішено застосувати одночасне поєднання цих типів навчання у вигляді моделі "Актор-Критик", щоб зекономити час затрачений на навчання та дати змогу мережі розуміти послідовність дій користувачів у довготривалій перспективі.

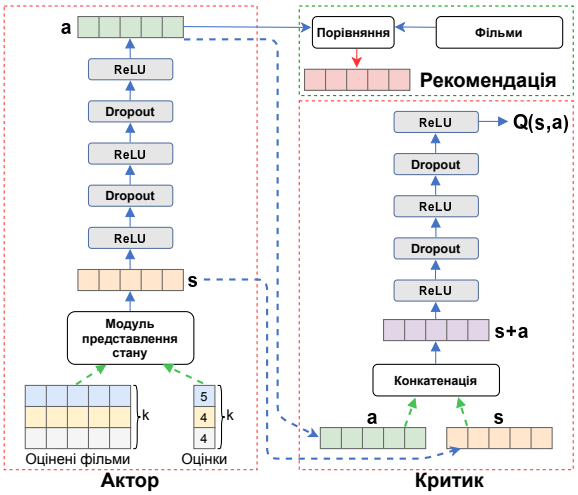

Рис. 4. Архітектура нейронних мереж "Актор" і "Критик"

Отже, система складається з двох мереж: мережі "Актор" і мережі "Критик" з архітектурою зображеною на рис. 4. За кожне з типів навчання відповідає своя мережа, відповідно для Policy-навчання – це мережа "Актор", а для Q-навчання – мережа "Критик". Мережа "Актор" генерує вибірку на основі останньої позитивної активності користувача та векторного подання фільмів з використанням "Модуля подання стану", про який буде сказано далі. Під позитивною мається на увазі активність, за якою користувач залишив позитивну оцінку фільму. Мережа "Критик" вибирає й оцінює наскільки успішною буде дана вибірка, і чи зацікавить вона користувача. Тобто, говорячи в загальному, мережа "Актор" генеруватиме вектор, який буде максимально схожим із рекомендованими фільмами у нашій вибірці на підставі певної характеристики (наприклад, скалярному добутку чи Евклідової відстані між цими векторами), які і будуть пропонуватися для рекомендації, а мережа "Критик" оцінюватиме наскільки така рекомендація доцільна в цей момент і обчислюватиме так зване Q-значення для поточного стану s_t та рекомендації a_t.

Маючи векторне подання фільмів та оцінки користувачів, можна перейти до навчання мережі. Спершу вся ця інформація потрапляє на мережу "Актор", яка генерує результуючий вектор, причому такої ж розмірності, що і вектор кожного окремого елемента вибірки ($n = 128$), який би був якомога ближче до векторів (рекомендацій), які якомога краще задовольняли б потреби користувача залежно від останніх k оцінок. Далі отриманий вектор порівнюють з усіма доступними в базі даних і знаходять найбільш відповідні для пропозиції користувачу з використанням певної характеристики (скалярного добутку, косинуса подібності, Евклідової відстані або інших метрик для порівняння векторів). Мережа "Критик" допомагає оцінити

роботу мережі "Актор" і викинути зайве зі списку рекомендацій. Також мережа "Критик" допомагає передбачити майбутню поведінку користувачів. Наприклад, система запропонувала користувачу певний фільм, він зверне на нього увагу, оцінить і система отримує високу винагороду. Але може трапитися подання, що користувачеві у майбутньому цей фільм дуже не сподобається і він змінить оцінку на нижчу, що в подальшому призведе до зниження винагороди. Отож усі майбутні дії користувача треба взяти до уваги. З цього прикладу можна легко побачити, що Policy-навчання не розраховане на довготривалий виграш і не здатне прогнозувати майбутню винагороду, що може виконувати Q-навчання. Тобто, рекомендації в кожен наступний момент мають бути побудовані так, аби користувач оцінив їх якомога вище, а система отримала якомога більшу винагороду від цієї взаємодії. Варто зауважити, що ці мережі, а також методи навчання були реалізовані з використанням мови програмування Python та бібліотеки Pytorch.

6. Модуль подання стану

Мережа "Актор", як це продемонстровано на рис. 4, містить "Модуль подання стану", який відіграє важливу роль для роботи двох мереж, бо його результат використовує кожна. Ознайомившись з [10], було вирішено використати схожий, але спрощений метод для подання стану користувача s_t. Для моменту t стан користувача може бути зображений у вигляді рівняння

$$s_t = f(U_t),$$

де f – функція модуля подання стану; $U_t = \{u_1, u_2, ..., u_k\}$ – векторне подання останніх k позитивних оцінок користувача, де $u_i, i = \overline{1,k}$ є одновимірним масивом елементів. Коли агент рекомендує новий елемент u_t і користувач оцінить його, то в наступний момент часу стан буде оновлено $s_{t+1} = f(U_{t+1})$, де $U_{t+1} = \{u_2, u_3, ..., u_k, u_t\}$, а в іншому випадку стан не зміниться $s_{t+1} = s_t$.

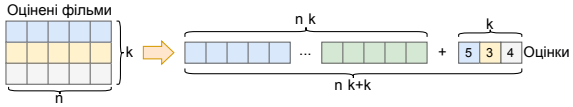

Рис. 5. Модуль подання стану користувача

Отже, спершу всі оцінки користувача за весь період сортуються у порядку спадання часу додавання, далі з цієї вибірки обираються перші k елементів, для яких і буде генеруватися рекомендація. Такий підхід обрано, бо важливішими є нещодавні взаємодії користувача з системою, а не всі за весь період часу, змінну k можна збільшувати або зменшувати залежно від потреб. На підставі останніх k оцінок обираються фільми з векторного подання фільмів, яким відповідають подані оцінки. Тобто отримується k векторів розмірності $n = 128$. Для отримання єдиного вектора всі k векторних подань об'єднуються між собою і додається вектор, який складається з числового подання відповідних оцінок. Тобто у підсумку отримується вектор розмірності $\dim(a_{input}) = k{\cdot}n{+}k$. Ця розмірність і буде дорівнювати кількості нейронів у вхідному шарі мережі "Актор", для мережі "Критик" $\dim(c_{input}) = \dim(a_{input}) + n$. Наприклад, якщо потрібно побудувати рекомендацію на підставі

останніх $k = 20$ оцінок і вектора ознак для кожного фільму $n = 128$, то вхідний шар мережі "Актор" буде містити $\dim(a_{input}) = 20 \cdot 128 + 20 = 2580$ нейронів, а вхідний шар мережі "Критик" $\dim(c_{input}) = 2580 + 128 = 2708$ нейронів. Для наочності роботу модуля подання стану зображено на рис. 5. Такий підхід використано через його простоту та зрозумілість, хоча не зменшуючи загальності, можна використовувати підходи на підставі скалярного добутку векторів оцінок та оцінених фільмів поданих в [10].

7. Архітектура мереж "Актор", "Критик" та алгоритм навчання

7.1. Мережа "Актор"

Мережа використовує Policy-навчання для побудови ймовірності вибору наступних дій (рекомендацій). Мережа "Актор" складається з трьох лінійних шарів (рис. 4) з активаційною функцією ReLu (лінійна випрямляюча функція), яка задається формулою $f(x) = \max(0, x)$ і двох шарів Dropout, які випадково занулюють 5% вхідних сигналів, що дає змогу запобігти перенавчанню мережі. Внаслідок роботи цієї мережі отримується вектор із $n = 128$ ознак.

Задача цієї мережі полягає у тому, що для заданого користувача в момент часу t вона генерує дію a_t на підставі його стану s_t. Спершу оцінки користувача $U_t = \{u_1, u_2, ..., u_k\}$ з його k останніх позитивних взаємодій із системою потрапляють в "Модуль подання стану", аби згенерувати підсумоване подання стану цього користувача $s_t = f(U_t)$. Далі після проходження декількох лінійних шарів з активаційною функцією ReLu та Dropout шарів стан s_t перетворюється в дію $a_t = p(s_t)$, яка є результатом роботи цієї мережі і вектором ознак розмірності $n = 128$, значення яких перебувають у межах (-1, 1). Навчання цієї мережі відбувається на підставі градієнта детермінованої стратегії [8], завдяки якому можна оновити ваги з використанням наступного градієнта

$$\nabla(p_0) \approx \frac{1}{N} \sum_t \nabla Q_0(s_t, p_0(s_t)) \nabla p_0(s_t).$$

У цьому випадку для мінімізації похибки буде використано метод оптимізації Адама [17]. Для побудови рекомендації згенерований так вектор порівнюється зі всіма векторами фільмів і заходять m найближчих до нього, де m – це кількість елементів, які треба порекомендувати. Це порівняння можна зробити з використанням скалярного добутку $rank_i = u_i \cdot a_t^T, \quad i = \overline{1, N}$ і тоді зі списку всіх фільмів розмірності N вибирають m найближчих за значенням.

7.2. Мережа "Критик"

Мережа використовує Q-навчання і застосовується для того, щоб оцінити наскільки хорошою буде винагорода з цього стану s та рекомендації a і визначається функцією Q(s, a). Вхід у мережу "Критик" – це поєднання стану користувача s згенерованого "Модулем подання стану" та дії a (наближеної рекомендації) згенерованої мережею "Актор". Виходом цієї мережі є так зване Q-значення подане у вигляді скалярного значення. Відповідно до обчисленого Q-значення ваги мережі "Актор" оновлюються для поліпшення винагороди від цієї наближеної рекомендації a. Якщо навчання мережі "Актор" відбувається з використанням градієнта

детермінованої стратегії, то мережа "Критик" оновлюється за рахунок мінімізації використанням методу Адама [17] середньоквадратичної похибки, яка визначається так:

$$E = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q_0(s_i, a_i))^2, y_i = r_i + \gamma Q'(s_{i+1}, p'(s_{i+1})), \quad i = \overline{1, N}.$$

Ця мережа складається з трьох лінійних шарів, як це зображено на рис. 4, з активаційною функцією ReLu і двох шарів Dropout, які і для мережі "Актор" занулюють 5% вхідних сигналів для запобігання перенавчанню мережі.

Завдання цієї мережі полягає в тому, що потрібно підібрати якомога кращий результат для рекомендації користувачеві, такий, який би давав якомога більшу винагороду, тобто, аби користувач виявив зацікавленість у цій пропозиції, звернув на неї увагу, вподобав чи поставив якусь оцінку в майбутньому.

Алгоритм 1: Алгоритм навчання мереж «Актор» та «Критик»

Вхідні дані: початкова вага для мережі «Актор» p_0, початкова вага для мережі «Критик» Q_0, коефіцієнт знижки γ, розмір вибірки N, к-сть позитивних взаємодій із системою k, функція винагороди R.

Випадковим чином ініціалізуємо навчальну мережу «Актор» p_0 та навчальну мережу «Критик» Q_0 з вагами θ та ω відповідно
Ініціалізуємо цільову мережу «Актор» p' та цільову мережу «Критик» Q' з вагами θ' та ω', причому $\theta \to \theta'$, $\omega \to \omega'$
for $session = \overline{1, M}$ **do**
  for $t = \overline{1, T}$ **do**
    Знайти поточний стан $s_t = f(U_t)$, де $U_t = \{u_1, u_2, ..., u_k\}$ з допомогою «Модуля представлення стану»
    Знайти дію $a_t = p(s_t)$ відповідно до поточного стану
    Порекомендувати елемент u_t відповідно до дії a_t з допомогою оцінки $rank = u_t \cdot a_t^T$
    Обчислити винагороду $r_t = R(s_t, a_t)$ на основі відгуку користувача
    Обчислити новий стан $s_{t+1} = f(U_{t+1})$, де $U_{t+1} = \{u_2, u_3, ..., u_k, u_t\}$
    if $r_t > 0$ **then**
      $U_{t+1} = \{u_2, u_3, ..., u_k, u_t\}$
    else
      $U_{t+1} = U_t$
    end if
    Генеруємо вибірку $(s_i, a_i, r_i, s_{i+1}), i = \overline{1, N}$ на основі переходу (s_t, a_t, r_t, s_{t+1})
    Обчислюємо $y_i = r_i + \gamma Q'(s_{i+1}, p'(s_{i+1}))$, де $i = \overline{1, N}$
    Оновлюємо ваги мережі «Критик» з допомогою мінімізації середньої квадратичного похибки $E = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q_0(s_i, a_i))^2$
    Оновлюємо ваги мережі «Актор» з використанням градієнту $\nabla(p_0) \approx \frac{1}{N}\sum_t \nabla Q_0(s_t, p_0(s_t))\nabla p_0(s_t)$
    Оновлюємо ваги цільових мереж $\tau\theta + \theta'(1-\tau) \to \theta'$ та $\tau\omega + \omega'(1-\tau) \to \omega'$ для мереж «Актор» і «Критик» відповідно
  end for
end for

Рис. 6. Навчання двох мереж

7.3. Алгоритм навчання нейронних мереж

У процесі навчання з використанням алгоритму градієнта глибокої детермінованої стратегії (англ. Deep Deterministic Policy Gradient) [8] використовується концепція цільової та навчальної мереж. Цільова мережа стабільніша, ніж навчальна, бо оновлюється з використанням навчальних параметрів у функції програмного оновлення. Така концепція краще навчається та демонструє кращі показники в

Таблиця 1

Точність мереж залежно від кількості ітерацій

Епох навчання	Точність "Актор"	Точність "Критик"	Час навчання (хв)
100	65.77%	63.86%	01:46
250	83.13%	79.78%	02:52
500	91.62%	87.21%	04:20
1000	95.62%	95.45%	08:04
2500	98.43%	98.76%	18:20
5000	98.94%	98.83%	37:40

цілому. Алгоритм навчання двох мереж зображено у вигляді псевдокоду **Алгоритм 1**.

8. Приклад навчання та застосування

Навчання нейронної мережі було проведено на 10 тисячах фільмів та 1 млн відгуків користувачів. Тестування та навчання системи відбувалось на процесорі $IntelCore^{\text{TM}}i7-9700K$ з тактовою частотою $4.8GHz$, який містить 8 ядер та 8 потоків з 64-розрядною ОС Ubuntu 20.04 та 32 Gb оперативної пам'яті. Параметри навчання згідно з Алгоритмом 1 такі:

$$\gamma = 0.9,\ \tau = 0.01,\ \theta = 0.5,\ \theta' = 0.1,\ \omega = 0.5,\ \omega' = 0.01.$$

Для навчання було обрано різну кількість циклів, як видно з табл. 1. Для порівняння й оцінки точності моделі було відібрано випадково 5% усіх відгуків. Точність кожної з мереж і загальний час затрачений на навчання також подано у табл. 1, з якої видно, що збільшення циклів навчання більше тисячі призводить до поліпшення лише на декілька відсотків, проте суттєво збільшує час навчання, тому 1000 циклів навчання є оптимальним у цьому випадку. Різницю між навчальною та тестовою вибіркою для 1000 епох навчання можна побачити на рис. 7. Точність мережі "Актор" становила 95.62%, мережі "Критик" – 95.45%, а затрачений час 08 хв 04 секунд. Такий швидкий час навчання варто пов'язувати передусім з потужним апаратним забезпеченням ПК, на якому відбувалось навчання.

9. Висновки

Ми запропонували підхід з використання машинного навчання з підкріпленням і моделі навчання "Актор-Критик" для побудови рекомендацій, який поєднує довготривалу побудову результуючої пропозиції та зосереджений на максимізації винагороди від взаємодії користувачів із системою. Продемонстрований приклад навчання та тестування на основі фільмів виявив велику точність і довів, що застосування такого підходу є вдалим і перспективним на відміну від інших статичних методів, адже на відміну від них пропонує рекомендацію як послідовний процес прийняття

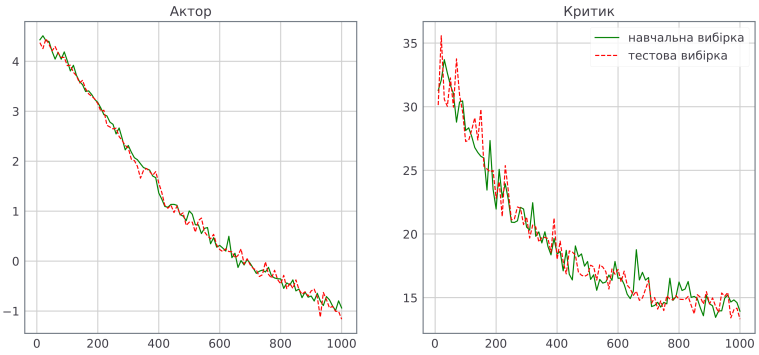

Рис. 7. Результат навчання нейронних мереж для 1000 циклів

рішень. Також такий підхід може бути використаний не лише в некомерційних цілях, а й великими компаніями для рекомендації своєї продукції, її поліпшення, збільшення прибутків і задоволеності клієнтів.

Список використаної літератури

1. Chi Wai Lau News Recommendation System Using Logistic Regression and Naive Bayes Classifiers / Chi Wai Lau // Citeseer. – 2011.
2. Hejazinia M. Accelerated learning from recommender systems using multi-armed bandit / M. Hejazinia, K. Eastman, S. Ye, A. Amirabadi, R. Divvela / arXiv. – 2019.
3. Karani D. Electronic Sources: Introduction to Word Embedding and Word2Vec / D. Karani – 2018, – Available from: https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa
4. Kawale J. Electronic Sources: A Multi-Armed Bandit Framework for Recommendations at Netflix / J. Kawale, E. Chow. – 2016. – Available from: https://info.dataengconf .com/hubfs/SF 18. – Slides/DS/A Multi-Armed Bandit Framework for Recommendations at Netflix.pdf
5. Koehrsen W. Electronic Sources: Neural Network Embeddings Expained / W. Koehrsen – 2018. – Available from: https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526
6. Koren Y. Matrix factorization techniques for recommender systems / Y. Koren, R. M. Bell, C. Volinsky / IEEE Computer. – 2009. – Vol. 42, № 8. – P. 30–37.
7. Li L. A contextual-bandit approach to personalized news article recommendation / L. Li, W. Chu, J. Langford, R. E. Schapire // WWW 2010, Raleigh, North Carolina, USA, April 26–30. – 2010. – P. 661–670.
8. Lillicrap T.P. Continuous control with deep reinforcement learning / T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra // CoRR. – 2015. – Vol. abs/1509.02971.
9. Linden G. Amazon.com recommendations: Item-to-item collaborative filtering / G. Linden, B. Smith, J. York // IEEE Internet Computing. – 2003. – Vol. 7, № 1. – P. 76–80.
10. Liu F. Deep Reinforcement Learning based Recommendation with Explicit User-Item Interactions Modeling / F. Liu, R. Tang, X. Li, W. Zhang, Y. Ye, H. Chen, H. Guo, Y. Zhang // arXiv. – 2019.

11. Mooney R.J. Content-based book recommending using learning for text categorization / R.J. Mooney, L. Roy // ACM DL. – 2000.
12. 55Mustansar Ali Ghazanfar, Adam Prugel-Bennett Ghazanfar M.A. "An Improved Switching Hybrid Recommender System Using Naive Bayes Classifier and Collaborative Filtering / Mustansar Ali Ghazanfar, Adam Prugel-Bennett // ResearchGate. – 2010.
13. O'Sullivan C. Electronic Sources: A Step-By-Step Introduction to PCA / C.O'Sullivan. – 2018. – Available from: https://towardsdatascience.com/a-step-by-step-introduction-to-pca-c0d78e26a0dd
14. Park J. Electronic Sources: Word Embedding in NLP: One-Hot Encoding and Skip-Gram Neural Network / J. Park. – 2020, – Available from: https://towardsdatascience.com/word-embedding-in-nlp-one-hot-encoding-and-skip-gram-neural-network-81b424da58f2
15. Saket G. Electronic Sources: Contextual multi-armed bandit – (Intuition behind Netflix Artwork Recommendation) / G. Saket. – 2020. – Available from: https :// medium.com /analytics-vidhya /contextual-multi-armed-bandit-intuition-behind-netflix-artwork-recommendation-b221a983c1cb
16. Sutton R. Reinforcement Learning: An Introduction / R. Sutton, A. Sutton. – Cambridge, Massachusetts. London, England: A Bradford Book, 2 edition, 2015. – 338 p.
17. Tong Q. Calibrating the Adaptive Learning Rate to Improve Convergence of ADAM / Q. Tong, G. Liang, J. Bi // arXiv. – 2019.
18. Zeng C. Online context-aware recommendation with time varying multi-armed bandit / C. Zeng, Q. Wang, S. Mokhtari, T. Li // SIGKDD, San Francisco, CA, USA, August 13-17. – 2016. – P. 2025–2034.
19. Zhao Z. Deep Reinforcement Learning for List-wise Recommendations / Z. Zhao, L. Roy // arXiv. – 2019. – P. 2–9.

Стаття: надійшла до редколегії 06.10.2021
доопрацьована 03.11.2021
прийнята до друку 24.10.2021

RECOMMENDATION SYSTEM DEVELOPMENT USING REINFORCEMENT LEARNING

B. Romaniuk, O. Peliushkevych, Y. Shcherbyna

Ivan Franko National University of Lviv,
Universytetska Str, 1, Lviv, 79000,
e-mail: bohdan2307@gmail.com, olga.peliushkevych@lnu.edu.ua,
yuriy.shcherbyna@lnu.edu.ua

Nowadays the recommendation is an important part of human life, it is used for industrial, business, and even academic purposes. Currently, various recommendation techniques are provided such as content-based filtering [11], matrix factorization [6], Naive Bayes classifier [1, 12], logistic regression [1], and multi-armed bandit approaches [7, 18]. Unfortunately, all of them are considering the recommendation as a static process, assuming that user preferences are not changing during the period of time and they are not taking into consideration that user interactions with the system are sequential. These methods and algorithms are widely used by giant companies like Amazon [9], Netflix [4, 15] and Google [2]. The article demonstrates using machine learning and especially reinforcement learning for items recommendation based on positive interactions of users with the system. This approach, unlike others, is dynamic, otherwise, it is adapted to user preferences changes and allows to focus on maximizing the benefits of user interaction with recommendation system. Besides that reinforcement learning also is focused on long-term reward and understands that user interactions with the system are sequential. The article describes an example of using this approach to recommend movies based on user positive ratings

provided to that movies, but speaking in general this technique can also be used for any type of information the only necessary thing is a representation of these elements to work with the neural network interface.

The environment of the proposed recommendation system is represented as Markov Decision Process [19] in which the user interacts with the recommendation agent and that agent creates proposed items for a specific user. Neural network architecture is built using the "Actor-Critic" learning conceptual model [19] which combines two methods of learning: Policy-learning as of "Actor" network and Q-learning as of "Critic" network. "Actor" network is taking users' positive interaction with the system as an input and generates approximate recommendation. On the other hand "Critic" is taking that approximate recommendation, concatenate it with positive interaction, and calculate Q-value to determine how good will be that approximate recommendation. The combination of these two learning techniques considers both dynamic adaptations to user preferences changes and long-term interactions which lead to fast performance and better recommendations are proposed. The main purpose of this process is to maximize reward, which means that the proposed recommendation will be positive for both user and system. Inside the system, movies are represented as an array of numeric elements to work properly with the neural network interface. Movies representation are created according to specific characteristics of each one, such as year of production, director, genres, country of production, companies, rating, popularity, budget, etc. To combine all those properties principal component analysis is used. Besides that to represent user ratings there was developed "State representation module", which combines rated movies and numeric rates itself and represents them in the same way as an array of digits.

10 thousand movies and 1 million rates from a dataset called MovieLens were used to train the model, from which 95% of ratings were used to train the model and the other 5% for accurate calculation. The proposed approach demonstrates great convergence up to 98% on big data sets and does not take a big amount of time to train the neural network model.

The proposed methodology combines the dynamic approach of generating recommendations understanding user actions as a sequential process and concentrating on getting a long-term reward. The recommendation in such a way is very perspective in the future and can be used by big companies to maximize their profits from sealing different types of things or information.

Key words: machine learning with reinforcement, neural network, presentation of Markov's decision-making process, recommendation system, Q-training, Police training.