УДК 004.89 **DOI:** 10.20535/2523-4455.2017.22.5.105016

Оптимізація нейронної мережі алгоритму DeepStack для гри у Leduc Hold'em

Дорогий Я. Ю., к.т.н., доц., ORCID 0000-0003-3848-9852
e-mail argusyk@gmail.com
Цуркан В. В., к.т.н., ORCID 0000-0003-1352-042X
e-mail v.v.tsurkan@gmail.com
Лісовий В. Ю., ORCID 0000-0003-1694-8130
e-mail s.vlad96@gmail.com
Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського" kpi.ua
Київ, Україна

***Реферат*—В статті розглянуте питання реалізації нейронної мережі та підбору її структури, яка використовується в алгоритмі DeepStack. Наведений детальний опис алгоритму та принципу його роботи. Розглянутий алгоритм використовується для прийняття рішення під час гри в покер. Покер представлений як гра з неповною інформацією. Розрахунок стратегії відбувається на основі двох параметрів – контрфактичних значень опонента та діапазону гравця. Запропонована нейронна мережа використовується для розрахунку стратегії, а саме контрфактичних значень опонента. В якості нейронної мережі була вибрана мережа прямого розповсюдження. В якості даних для навчання використовувався набір вирішений покерних ситуацій, який включав в себе різні величини ставок та комбінації рук. Розглянуто декілька структур мереж та вибрана оптимальна. Критерієм вибору слугує оцінка вразливості стратегії.**

Бібл. 13, рис. 9.

Ключові слова — нейронна мережа; покер; стратегія; контрфактичні значення; дерево передбачення.

I. Вступ

Штучний інтелект за останні роки доволі швидко розвивається та використовується в різних сферах. Однією з таких сфер є ігри. В основному дана технологія застосовується для ігор з повною інформацією. Покер є прикладом гри з неповною інформацією, яка порушувала немало проблем в області штучного інтелекту. Недавно був розроблений алгоритм DeepStack, який зміг перемогти команду професійних гравців в Техаському Холдемі, які були спеціально запрошені Міжнародною федерацією покеру.

II. Аналіз літературних даних і постановка проблеми

Штучний інтелект в покері використовується для багатьох цілей: побудови дерева прийняття рішень, передбачення майбутніх дій для гравця з певним набором карт, розрахунків значень потенційних рук противників. Більшість алгоритмів для гри в покер засновані на використанні статистичних даних про опонентів, з якими проводиться гра. Так, програма-бот Rembrant [1] представляє собою простий та ефективний алгоритм. На кожному з етапів торгівлі, базуючись на даних про дії гравців, приймається рішення щодо подальших дій в грі. Даний алгоритм ділиться на дві частини: фазу префлопу та фазу постфлопу. В фазі префлопу(етап гри перед здачею відкритих карт) розраховується група, в яку потрапляє стартова рука, збирається інформація про дії опонента і на основі цієї інформації вибирається подальша дія боту. Алгоритм для гри в префлопі ділить можливі стартові руки на 9 груп. Кожна група представляє набір дій, які вибираються в залежності від дій опонента. Карти в групі згруповані на основі комбінацій і масті. Набір дій для однієї з груп представлений в табл. 1. Дана група представляє найсильніші стартові руки і відображає дві найвищі пари – два тузи і два королі. За таким принципом працюють інші вісім груп.

Табл. 1. Набір дій для групи 1, яка складається з двох найвищих пар

Суперник	Бот
Перша позиція, чек чи колл	Рейз 3-5х великих блайндів
Рейз	Рейз 3-4х малих блайндів
Ре-рейз	Ол-ін
Ол-ін	Колл

В фазі постфлопу(етап гри після здачі відкритих карт) розпочинається процес обрахунку і отримання кінцевої комбінації для поточного стану гри. Дана фаза ділиться на три частини – флоп, терн, рівер – і для кожної частини проводяться три цикли алгоритму: виклик функції стирання, яка видаляє інформацію про кінцеві комбінації попередньої частини фази, виклик функції порівняння відкритих карт і стартової руки для визначення поточної комбінації(пара, дві

пари, стрейт і т.д.), яка належить боту, виклик функції оцінки для визначення подальшої дії. Окрім цього, на етапі флопу збирається вся можлива інформація від противника для визначення стартової руки опонента. На основі зібраної інформації проводиться коригування стратегії бота на етапі терну і ріверу. Кінцеві комбінації поділяються на групи, як у фазі префлопу. Подальші дії розраховуються на основі зібраної інформації і поточної кінцевої комбінації. Для прикладу в табл. 2 наведений список дій для бота при кінцевій комбінації флеш.

ТАБЛ. 2. НАБІР ДІЙ ПРИ КІНЦЕВІЙ КОМБІНАЦІЇ ФЛЕШ

Суперник	**Дія бота при флеші з тузом**	**Дія бота при іншому флеші**
Рейз	Колл/Ре-рейз	Ре-рейз
Ре-рейз	Ол-ін/Колл	Ол-ін
Ол-ін	Колл	Колл/Фолд
Чек	Чек/Рейз	Чек/Рейз
Перша позиція, чек чи колл	Чек/Рейз	Чек/Рейз

Основним недоліком даного бота можна вважати негнучкість стратегії, оскільки професійний гравець в покер при достатньо довгій грі проти даної програми може викрити набори стратегій, якими користується Rembrant.

Доволі широкого розвитку набули алгоритми на основі генетичних алгоритмів. Так, в роботі [2] описується пошук найкращих агентів для гри в покер, які використовують еволюційну нейронну мережу для прийняття рішення. Автори використовують нейрону мережу прямого розповсюдження з трьома шарами: вхідний з 35 нейронами, прихований з 20 і вихідний з 3 нейронами. Вхідний шар призначений для оцінки поточного стану гри. Основні параметри, які надходять на вхід мережі є кількість фішок в загальному банку, кількість фішок для коллу, число опонентів, відсоток рук, які можуть виграти, кількість гравців між гравцем, чия черга робити хід і дилером, значення агресивності. Відсоток рук, які можуть виграти – величина, яка визначає ймовірність виграшу комбінації карт агента. Дана ймовірність залежить від сили руки, тобто ймовірності виграшу агента тільки з власною стартовою рукою, та потенціалу руки, тобто ймовірності виграшу агента з комбінацією, сформованою власними закритими і відкритими картами. Агресивність відображає стратегію ставок опонента, тобто допомагає передбачити можливу комбінацію противника. Вихідний вектор складається з трьох вузлів, які відображають фолд, колл та рейз. Вузол рейз при цьому ділиться на три складові, які відображають значення ставки (мала, середня, велика), що знаходиться за допомогою спеціальної таблиці розподілу величини ставок. Дана таблиця базується на статистичних спостереженнях. Функцією активації мережі виступала сигмоїд-функція.

Генерація самих агентів відбувається за рахунок генетичного алгоритму. Агенти конкурують один з одним, і найсильніші особи вибираються для розмноження і подальшої конкуренції. Загальна структура алгоритму представлена на рис. 1.

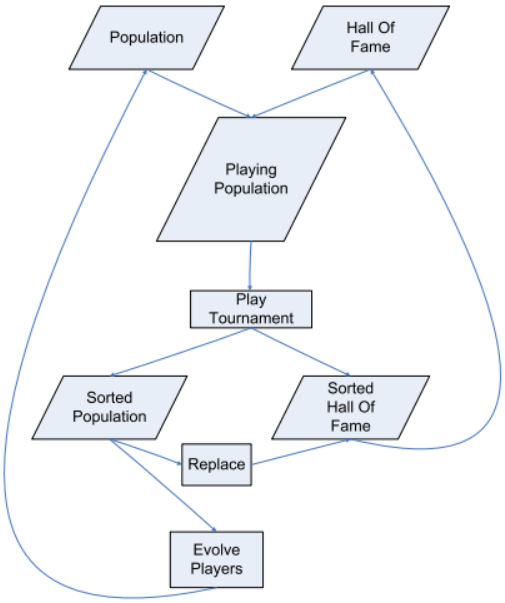

Рис. 1 Структура генетичного алгоритму

Після проведення нової генерації агентів отримана популяція проводить 1000 ігор в покер на турнірній основі для визначення найкращих гравців. Частина агентів, що виграла турнір, ставали потомками для наступної популяції агентів, інша ж частина відправлялася в зал слави. Зал слави використовується для тренування і визначення предків для нової популяції агентів. До недоліків даного підходу можна віднести також малу гнучкість отриманої стратегії, оскільки суперник може розгадати отриману стратегію і виникне необхідність перенавчання агента знову, що займає досить великий час.

Також штучний інтелект може використовуватися для прогнозування майбутніх дій противника. В роботі [3] представлена система, що формує власну стратегію гри на основі передбачення стратегії опонентів. Загальна структура роботи даного прикладу представлена на рис. 2. Раунд гри описується у вигляді дерева. Дерево реалізовано на основі абстрактної версії можливих ситуацій, які можуть виникати при поточному стані набору відкритих карт. Вузли в дереві побудовані для всіх можливих дій агента, що дозволяє алгоритму оцінки (пошуку в дереві) вибрати дію з найоптимальнішим результатом. Кінцеві вузли (leaf node) описують значення, що показує кількість фішок, які за прогнозами будуть виграні. Для вузлів, що відображають опонента(opponent action) за допомогою нейронної мережі розраховується ймовірність отримання тієї чи іншої комбінації цього самого опонента. На основі отриманих даних алгоритм формує ймовірнісну трійку, яка представляє собою розподіл дій (фолд, колл, рейз). Дана трійка показує, яку наступну дію можливо зробить опонент. Агент, знаючи розподіл майбутніх дій, має змогу розробити власну оптимальну стратегію.

Хоча дана система генерує кожен раз стратегію, яка є більш виграшною, ніж стратегія опонента, і з розглянутих рішень даний алгоритм є найбільш

комплексним і потужним, але при малій кількості даних про історію дій суперника передбачення буде мати доволі малу точність, а при великій кількості таких даних збільшується час на їх обробку.

Алгоритм, який описується в даній роботі, дозволяє використовувати доволі гнучку стратегію, яка перераховується на кожному етапі торгівлі, а нейрона мережа використовує невелику кількість даних для передбачення можливих подальших дій опонента.

III. Мета і завдання дослідження

Метою даної роботи є порівняння різних структур нейронної мережі, яка використовується в алгоритмі DeepStack, з метою отримання найменш вразливої стратегії.

IV. Опис алгоритму та експериментальні дослідження

A. *Алгоритм DeepStack*

DeepStack - універсальний алгоритм для великого класу розширених ігор з неповною інформацією [4]. Роботу даного алгоритму описують на прикладі покеру. Гру в покер можна розділити на три складові: “закритий” стан, який складається з двох карт одного з гравців, що невідомі іншим гравцям, і “відкритий” стан, що складається з відкритих карт, а також послідовність ставок, що здійснюються гравцями. Можливі послідовності відкритих карт в грі формують дерево з кожним відкритим станом, вузли якого, в свою чергу, формують інші піддерева з послідовностями інших відкритих станів (рис. 3).

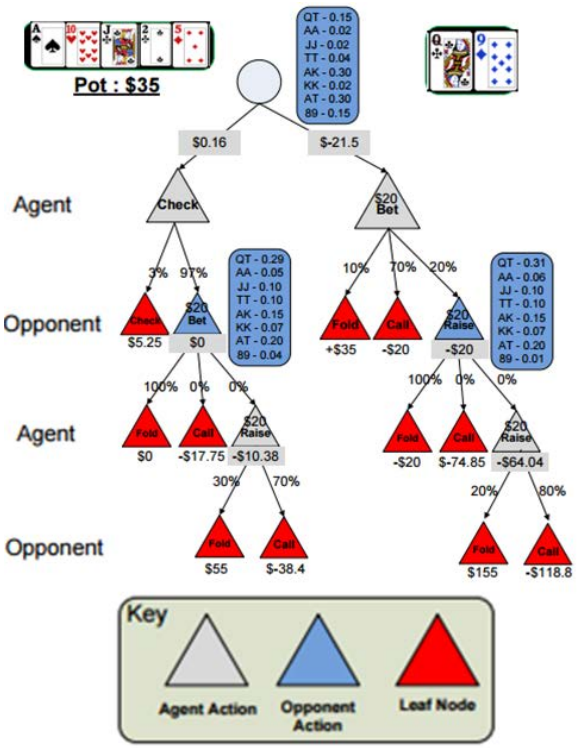

Рис. 2 Приклад дерева рішень

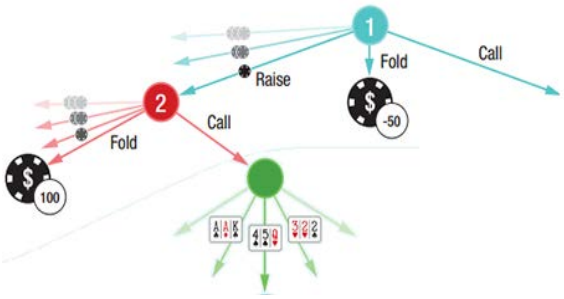

Рис. 3 Приклад частини дерева рішень, яке формує DeepStack

Стратегія гравця визначається як розподіл ймовірностей дій для кожної точки прийняття рішення, де точкою прийняття рішення є поєднання відкритих карт і руки самого гравця. З огляду на стратегію гравця, для будь-якого відкритого стану можна обчислити діапазон гравця як міру розподілу ймовірностей можливих рук гравця.

Алгоритм DeepStack можна розділити на три компонента: SLSC (sound local strategy computation) для поточного публічного стану, дерево передбачення з обмеженням на глибину та обмежений набір майбутніх дій.

Головною ідеєю алгоритму DeepStack є концепція continual re-solving, коли гравець кожен раз перераховує свою стратегію тільки на основі мінімальної інформації про те, як і чому він діяв, щоб досягти поточного відкритого стану.

Архітектуру алгоритму можна побачити на рис. 4.

DeepStack будує дерево гри, при цьому генеруючи ймовірність подальших дій для всіх карт, які можуть зберігатися у відкритому стані. Під час гри алгоритм використовує два вектори: власний діапазон гравця та контрфактичні значення його опонентів.

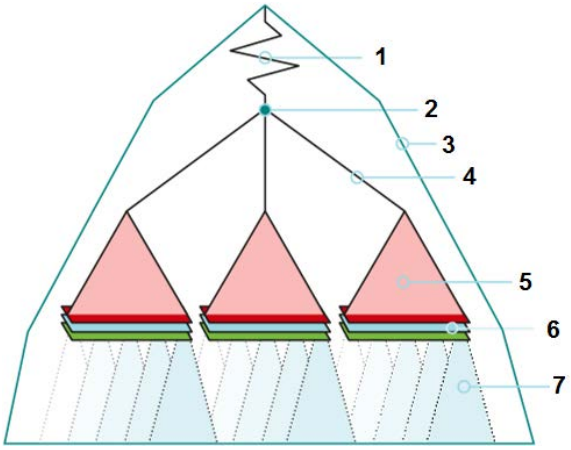

Рис. 4 Узагальнена архітектура DeepStack. 1 – відображає історію дій, 2 – поточний відкритий стан, 3 – ігрове дерево, 4 – можливі дії гравця, 5 – дерево передбачень, 6 – нейронна мережа, 7 – піддерево.

Під час гри власний діапазон оновлюється за правилом Баєса, використовуючи розраховані ймовірності дій, які вже були зроблені на попередньому кроці. Контрфактичні значення опонента обчислюються за допомогою алгоритму CFR. Для обрахунку ймовірності майбутніх дій, виконується процедура continual re-solving, яка використовує значення обох векторів. Передбачення дій обмежується кінцем раунду торгівлі. Під час процедури continual re-solving контрфактичні значення для відкритого стану за його межами апроксимуються за допомогою функції оцінки, яка представлена нейронною мережею. Нейронна мережа приймає на вхід відкритий стан та діапазон для поточного стану і на виході видає контрфактичні значення

B. Опис процедури continual re-solving

Припустимо, що гравець здійснив певні дії у відповідності до стратегії, але при досягнені певного відкритого стану забув цю стратегію. За допомогою continual re-solving можна відновити стратегію для нового піддерева без необхідності повторного проведення обчислень для всієї гри. Для цього гравцю необхідно знати власний діапазон та вектор очікуваних значень, отриманих опонентом на попередньому рішенні для кожної руки опонента. З даними значеннями гравець може реконструювати стратегію для решти гри. Кожне значення в векторі опонента є контрфактичним значенням, яке дає очікуване значення, якщо противник досягне відкритого стану з певною рукою. За допомогою алгоритму CFR визначають вектор контрфактичних значень опонента для будь-якого відкритого стану.

Застосування continual re-solving призводить до того, що певна стратегія використовується тільки в межах наступного ходу. А для того, щоб мати можливість відновити стратегію в відкритому стані, потрібно тільки відстежувати власний діапазон і відповідний вектором контрфактичних значень опонента.

На початку гри, діапазон є рівномірним і контрфактичні значення опонента для певного гравця ініціалізуються значеннями, які відповідають кожній закритій руці. Під час ходу відбувається процедура перерахунку піддерева для поточного відкритого стану з використанням збережених діапазону гравця та контрфактичних значень опонентів, і гравець діє щодо нової розрахованої стратегії. Після кожної дії відбувається оновлення діапазону гравця та контрфактичних значень, при чому діапазон оновлюється за допомогою нової стратегії та функції Баєса. Дана процедура ніколи не відслідковує діапазон противника, а лише його контрфактичні значення . Також дана процедура не потребує знання дій противника, що виділяє дане рішення серед традиційних процедур re-solving.

C. Поняття про контрфатичні значення та алгоритм CFR

Алгоритм CFR є модифікацією алгоритму узгодження за смутком [5].

В 2000 році Харт та Мас-Колел запропонували важливий алгоритм теорії ігор, який назвали узгодженням за смутком (англ. regret matching).

Для опису алгоритму опишемо деякі терміни з теорії ігор.

Алгоритм використовується для розширених ігор в нормальній формі з неповною інформаціє.

Гру в розширеній формі представляють у вигляді орієнтованого дерева, де кожна вершина відповідає ситуації вибору гравцем своєї стратегії.

Гра з неповною інформацією – гра, в якій учасники не знають всі ходи, які були зроблені до поточного моменту, а також не можуть знати можливі стратегії опонентів, що дозволяє передбачити майбутній хід гри.

Гра в нормальній формі представляється у вигляді кортежу $\{N, A, u\}$, де:

- $N = \{1 \dots, n\}$ – кінцева множина гравців;
- S_i – кінцева множина дій чи виборів і-того гравця;
- $A = S_1 \times \dots \times S_n$ – множина всіх можливих комбінації одночасних дій усіх гравців (кожна можлива комбінація одночасних дій називається профілем дії);
- u - функція відображення кожного профілю дій по вектору вигоди (виплат) для кожного гравця. Цю функцію також називають функцією корисності.

Ігри в нормальній формі відображаються у вигляді n-вимірних матриць або таблиць, де кожний вимір має свої рядки та стовпці, що відповідають діям одного гравця, а кожен перетин рядка та стовпця відповідає одному профілю дії.

Гру в нормальній формі називають грою з нульовою сумою, якщо сума значень вектору вигоди дорівнює нулю.

В іграх розширеної форми та з неповною інформацією виділяють наступні компоненти:

- Кінцева множина послідовностей дій H, яку називають історією. $Z \subseteq H$ називають історією терміналів (кінцевих вузлів). Тоді префікс $h \sqsubseteq z$ для $z \in Z$ називають історією нетерміналів (некінцевих вузлів), $h \in H/Z$.
- Функція P, яка присвоює кожній нетермінальній історії член $N \cup \{c\}$. P є функцією гравця. $P(h)$ – це гравець, який вибере дію після історії h. Якщо $P(h) = c$, тоді ймовірність визначає дію після історії h.
- Для кожного гравця $i \in N \cup \{c\}$ розподіл τ_i при $h \in H : P(h) = i$ з властивістю $A(h) = A(h')$ тоді, коли h та h' знаходяться в одному елементі розподілу. Для $I_i \in \tau_i$ позначимо через $A(I_i)$ множину $A(h)$та через $P(I_i)$ множину $P(h)$ для будь-якого $h \in I_i$. τ_i називають інформаційним розподілом і-того гравця, а множину $I_i \in \tau_i$ інформаційною множиною і-того гравця. Інформаційну множину можна уявляти як набір вузлів дерева.

Також для пояснення даного алгоритму використовують поняття ідеальної пам'яті, тобто при кожному виборі дії кожен гравець пам'ятає, що він робив на попередніх кроках гри, і кожен гравець пам'ятає все, що він знав раніше щодо гри.

Стратегія і-того гравця σ_i є функцією, яка визначає розподіл по $A(I_i)$ для кожного $I_i \in \tau_i$. Через $\sum_i$ позначають множину всіх стратегій і-того гравця. Профіль стратегії σ складається з стратегії для кожного гравця, $\sigma_1, \dots, \sigma_n$. Через σ_{-i} позначають стратегії в σ, виключивши σ_i. Нехай $\pi^\sigma(h)$ – ймовірність історії h, якщо всі гравці вибирають дії, що відповідають σ. Можна розкласти $\pi^\sigma(h) = \prod_{i \in N \cup \{c\}} \pi_i^\sigma(h)$ як вклад кожного гравця в цю ймовірність. Тоді $\pi_i^\sigma(h)$ – ймовірність того, що, якщо і-тий гравець грає у відповідності до σ, то для всіх історій h', які є префіксом h з $P(h) = i$, і-тий гравець застосує відповідну дію в h. Нехай $\pi_{-i}^\sigma(h)$ буде добуток внеску всіх гравців, за винятком і-того гравця. Для $I \subseteq H$ визначимо $\pi^\sigma(I) = \sum_{h \in I} \pi^\sigma(h)$, як ймовірність досягнення конкретної інформаційної множини при заданому профілю стратегії σ, причому $\pi_i^\sigma(I)$ та $\pi_{-i}^\sigma(I)$ визначені аналогічним чином. Нехай $\pi^\sigma(h, z) = \pi^\sigma(z)/\pi^\sigma(h)$, якщо $h \sqsubseteq z$ або нуль в іншому випадку. Значення $\pi_i^\sigma(h, z)$ та $\pi_{-i}^\sigma(h, z)$ визначаються аналогічно. Очікуваний виграш і-того гравця тоді буде $u_i(\sigma) = \sum_{h \in Z} u_i(h)\pi^\sigma(h)$.

З огляду на профіль стратегії σ, можна визначити найкращу відповідну стратегію гравця як стратегію, яка максимізує його виграш при умові, що інші гравці грають у відповідності до σ.

Величину ϵ - називають наближенням рівноваги Неша, тобто це профіль стратегії σ, що задовольняє умові:

$$\forall i \in N \; u_i(\sigma) + \epsilon \geq \max_{\sigma_i' \in \sum_i} u_i(\sigma_i', \sigma_{-i}). \qquad (1)$$

Якщо $\epsilon = 0$, тоді σ називають рівновагою Неша: жоден гравець не має будь-яких причин до відхилення стратегії, оскільки всі гравці грають найкращу відповідну стратегію.

Нехай σ_i^t стратегія, яку використовує і-тий гравець в раунді t. Середнє загальне значення смутку для і-того гравця в часі T буде визначатися як:

$$R_i^T = \frac{1}{T} \max_{\sigma_i^* \in \sum_i} \sum_{t=1}^{T} (u_i(\sigma_i^*, \sigma_{-i}^t) - u_i(\sigma^t)). \qquad (2)$$

Крім того, визначимо $\bar{\sigma}_i^t(a|I)$ як середню стратегію і-того гравця в діапазоні від 1 до T. Для кожної інформаційної множини $I \in \tau_i$ та для кожного $a \in A(I)$:

$$\bar{\sigma}_i^t(a|I) = \frac{\sum_{t=1}^{T} \pi_t^{\sigma^t}(I)\sigma^t(a|I)}{\sum_{t=1}^{T} \pi_t^{\sigma^t}(I)}. \qquad (3)$$

Існує зв'язок між значенням смутку, середніми стратегіями і рівновагою Неша – в іграх з нульовою сумою якщо $R_{i \in \{1,2\}}^T \leq \epsilon$, тоді $\bar{\sigma}^T$ є 2ϵ рівновагою.

Алгоритм для вибору σ_i^t для і-того гравця зводить значення смутку до мінімуму, якщо середнє загальне значення смутку і-того гравця (незалежно від σ_{-i}^t) прямує до нуля, коли t прямує до нескінченності. В результаті, алгоритми мінімізації смутку можна використовувати в якості методу обрахунку наближеної рівноваги Неша.

Концепцію алгоритму CFR у загальному випадку автори [6] описують через розкладання загального значення смутку в ряд адитивних членів значень смутку, які потім мінімізуються незалежно.

Нехай і-тий гравець вибрав певну дію з однієї інформаційної множини $I \in \tau_i$. Тоді $u_i(\sigma, h)$ – очікувана корисність, враховуючи, що всі гравці використовували стратегію σ і історія h була досягнута. Контрфактична корисність $u(\sigma, I)$ – очікувана корисність, яка враховує, що інформаційна множина I була досягнута і що всі гравці використовували стратегію σ, за виключенням і-того гравця, який грав до досягнення інформаційної множини I. Це значення можна розглядати як міру досягнення і-тим гравцем деякої інформаційної множини I при умові, що гравець намагається її досягнути. Для всіх $a \in A(I)$ визначають $\sigma|_{I \to a}$, що є профілем стратегії, який ідентичний σ, за виключенням того, шо і-тий гравець завжди вибирає дію a, коли вона належить інформаційній множині I. Пряме значення контрфактичного смутку дорівнює:

$$R_{i,imm}^T(I) = \frac{1}{T} \max_{a \in A(I)} \sum_{t=1}^{T} \pi_{-i}^{\sigma^t}(u_i(\sigma^t|_{I \to a}, I) - u_i(\sigma^t)). \qquad (4)$$

Фактично, дана формула показує значення смутку гравця в його рішеннях при деякій інформаційній множині I з точки зору контрфактичної корисності, з додатковим ваговим коефіцієнтом для контрфактичної ймовірності, яка відображає, що і-тий гравець досягне деякої інформаційної множини I в даному раунді при умові, що гравець намагається її досягнути. Оскільки позитивне значення смутку відображає неефективність стратегії, тоді нехай $R_{i,imm}^{T,+}(I) = \max(R_{i,imm}^T(I), 0)$ буде додатною частиною прямого значення контрфактичного смутку. Тоді

$$R_i^T \leq \sum_{I \in \tau_i} R_{i,imm}^{T,+}(I). \qquad (5)$$

Доведення даної формули представлено у роботі [5].

Оскільки мінімізація прямого контрфактичного значення смутку мінімізує загальне значення смутку, це дозволяє знайти наближену рівновагу Неша, якщо можна мінімізувати тільки пряме контрфактичне значення смутку.

Ключовою особливістю прямого контрфактичного значення смутку є те, що його можна мінімізувати, контролюючи тільки $\sigma_i(I)$. Для цього можна використати алгоритм Блеквела, який дозволяє мінімізувати це значення незалежно від кожної інформаційної множини. Зокрема, для всіх $I \in \tau_i$ та $a \in A(I)$:

$$R_i^T(I,a) = \frac{1}{T}\sum_{t=1}^{T} \pi_{-i}^{\sigma^t}(u_i(\sigma^t|_{I\to a}, I) - u_i(\sigma^t)). \quad (6)$$

Визначимо $R_i^{T,+}(I,a) = \max(R_i^T(I,a), 0)$, тоді стратегія для моменту часу T+1 буде:

$$\sigma_i^{T+1}(I)(a) = \begin{cases} \frac{R_i^{T,+}(I,a)}{\sum_{a\in A(I)} R_i^{T,+}(I,a)}, \sum_{a\in A(I)} R_i^{T,+}(I,a) > 0 \\ \frac{1}{|A(I)|}, \text{в усіх інших випдках.} \end{cases} \quad (7)$$

Іншими словами, дії вибираються пропорційно кількості позитивних контрфактичних значень смутку, які не були вибрані. Якщо ніякі дії не мають позитивного контрфактичного значення смутку, тоді дія вибирається випадковим чином.

Якщо і-тий гравець вибирає дії відповідно до формули (7), тоді $R_{i,imm}^T(I) \le \Delta_{u,i}\frac{\sqrt{|A_i|}}{\sqrt{T}}$, отже $R_{i,}^T \le \Delta_{u,i}\frac{\sqrt{|A_i|}}{\sqrt{T}}$, де $|A_i| = \max_{h:P(h)=i}|A(h)|$.

Доведення даної формули представлено у роботі [5].

Цей результат встановлює, що стратегія в (7) може використовуватися для обчислення рівноваги Неша. Крім того, оцінка середнього загального значення смутку лінійна за кількістю наборів інформації.

D. *Експериментальні дослідження*

Нейрона мережа розроблена на мові Lua з використанням фреймворку torch. В якості імплементації використовувалась реалізація [12]. Був проведений експеримент для отримання оптимальної структури нейронної мережі, за допомогою якої алгоритм згенерує найменш вразливу стратегію. Оскільки звичайний Техаський Холдем доволі громіздка гра, для експерименту використовувалась модифікація покеру Leduc holdem.

Leduc holdem – модифікація покеру, яка використовується в наукових дослідженнях(вперше представлена в [7]). Гра проводиться колодою з шести карт, що включає в себе дві масті з трьох звань (туз, король, і королева). Гра починається з того, що кожному гравцю видають по одній закритій карті і починається раунд торгівлі. Наступна карта розігрується для всього столу і є відкритою і проводиться ще один раунд торгівлі. Якщо карта гравця має теж звання, що і відкрита карта на столі, то цей гравець виграє. В іншому випадку, виграє той гравець, який має більше звання власної карти.

E. *Архітектура нейронної мережі*

В даному експерименті використовувалася звичайна мережа прямого розповсюдження з різною кількістю прихованих шарів: 2 та 4.

В роботі [4] структурно мережа мала 7 прихованих шарів. Навчання мережі проводилось за допомогою графічного процесору відеокарти, який значно пришвидшує необхідні розрахунки. Оскільки при дослідженні не було можливості використовувати графічний процесор, тому для зменшення часу на обрахунок було вирішено спроектувати мережі з кількістю шарів, яка приведена вище.

В якості функцій активації використовувалися:

- Тангенс гіперболічний (tanh) - $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Графік даної функції представлений на рис.5.
- ReLU – функція випрямленої прямої одиниці (rectified linear unit); $f(x) = \max(0, x)$. Графік даної функції представлений на рис.6.
- SoftPlus – гладке наближення до функції ReLU; $f_i(x) = \frac{1}{\beta} * \log(1 + e^{\beta * x_i})$. Графік даної функції представлений на рис.7.
- PReLU – параметрична функція ReLU, параметр якого залежить від нахилу негативної частини; $f(x) = \max(0, x) + a * \min(0, x)$. Графік даної функції представлений на рис.8.

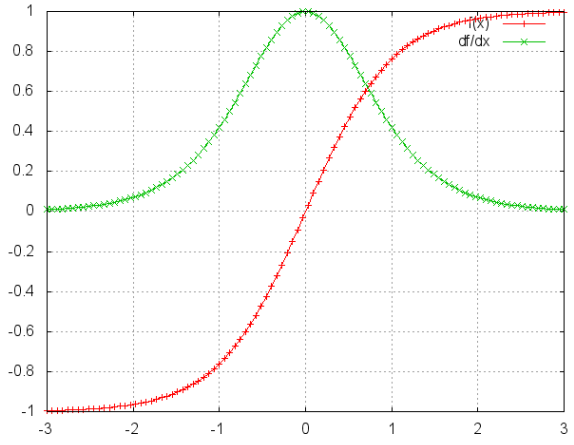

Рис. 5 Функція активації tanh

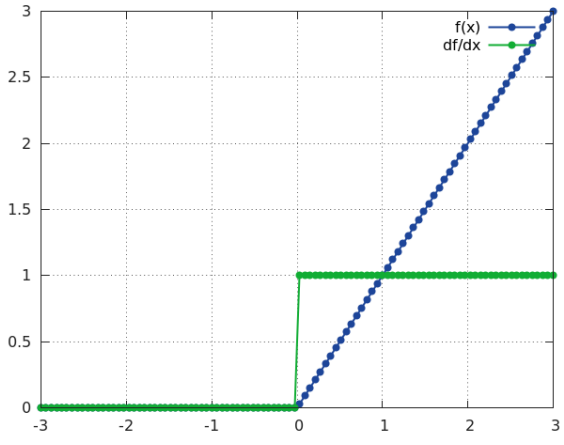

Рис. 6 Функція активації ReLU

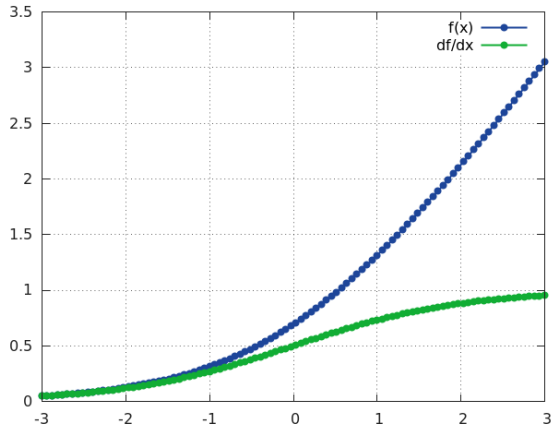

Рис. 7 Функція активації SoftPlus

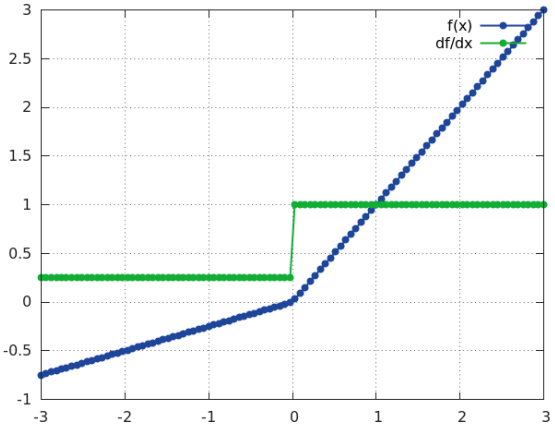

Рис. 8 Функція активації PReLU

Оскільки для експерименту використовувалися мережі невеликої глибини і гостро не стояла проблема зникнення градієнту, то була вибрана функція активаціїї tanh.

Вибір функції ReLU обгрунтований тим, що це стандартний вибір при побудові мереж, бо вона доволі легко розраховується і показує гарні статистичні результати.

Функції SoftPlus і PReLU є кращими, ніж ReLU, тому що вони попереджують розрідження вагів нейронів [13], що дозволяє градієнту безперешкодно рухатися під час зворотного розповсюдження помилки.

Функція SoftPlus є більш чуттєвою до змін у вагах біля значення 0, що позитивно впливає на процес початку навчання, коли усі ваги ініціалізовані близькими до нуля значеннями.

F. Тренування нейронної мережі

Мережі були навчені за рахунок 1000000 вирішених випадкових покерних ситуацій, які були згенеровані раніше. В роботі 600000 ситуацій було використано для навчання, 400000 для тестування. Мережі пройшли навчання з використанням градієнтної процедури оптимізації спуску Адама [8] з втратою Хьюбера [9].

Оцінка оптимальності отриманої структури мережі проводилась через оцінку вразливості стратегії в другому раунді Leduc holdem. Вразливість стратегії - різниця між контрфактичним значенням даного відкритого стану і очікуваної корисності під рівновагою Неша. Величиною для оцінки вразливості стратегії може слугувати mili big blind per hands (mbb/h)[10] – середнє значення великих блайндів на 100 роздач. Ця величина показує кількість фішок, що програє гравець, притримуючись певної стратегії. Значення вразливості стратегії відображається в табл 3.

ТАБЛ. 3. ЗНАЧЕННЯ ВРАЗЛИВОСТІ СТРАТЕГІЇ

Мережа	**Значення вразливості стратегії(mbb/h)**
З 2 прихованими шарами, функція активації Tanh	18.342666625977
З 2 прихованими шарами, функція активації ReLU	16.371120452881
З 2 прихованими шарами, функція активації SoftPlus	16.97642230987
З 2 прихованими шарами, функція активації PReLU	16.55432336609
З 4 прихованими шарами, функція активації Tanh	18.174759864807
З 4 прихованими шарами, функція активації ReLU	19.985319137573
З 4 прихованими шарами, функція активації SoftPlus	15.709207057953
З 4 прихованими шарами, функція активації PReLU	17.560428046881

Як можна спостерігати з табл. 3, найкращим результатом виявилась мережа з 4 прихованими шарами та функцією активації SoftPlus. В роботі [4] та імплементації [12] в якості такої функції використовувалась PReLU. Причину отримання кращого результату з використанням функції SoftPlus відносно PReLU можна пояснити наступним: при ініціалізації вагів їх значення близькі до нуля, а оскільки дана функція є більш гладкою і в значені 0 її похідна є неперервною, на відмінну від PReLU, що можна спостерігати з рис. 7 та 8 відповідно, це призвело до більш точнішого розрахунку контрфактичних значень опонента та стратегії в подальшому.

Найкращий результат, а саме стратегію, яка розроблена на основі використання нейронної мережі з 4 прихованими шарами та функцією активації SoftPlus, можна порівняти з результатами, які були отримані за допомогою інших алгоритмів. Дане порівняння представлене на рис. 9. Для порівняння використовуються алгоритм з роботи [5] (умовне позначення на гістограмі Program 1), програма з роботи [11] (умовне позначення Program 2) та система з роботи [3] (умовне позначення Program 3).

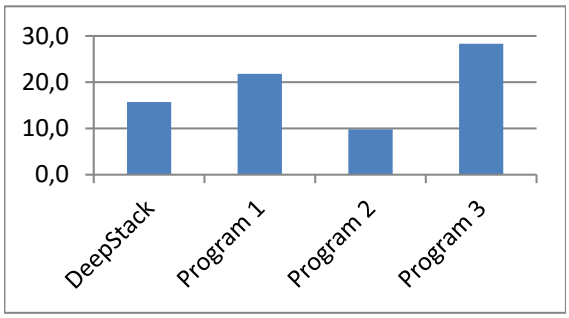

Рис. 9 Гістограма порівняння отриманих результатів для алгоритму DeepStack та інших алгоритмів

Як видно з рис. 9, отримане значення вразливості стратегії має середні результати в порівнянні з іншими алгоритмами. Різниця між найкращим для даного порівняння результатом та вразливістю стратегії DeepStack складає приблизно 30%. Причиною такої різниці може слугувати те, що автори праці [11] використовували для розрахунку алгоритм CFR+, що є модифікацією CFR, який використовується в DeepStack.

Різниця ж між іншими результатами, приведеними на рис. 9 відображає ефективність роботи представленого алгоритму і доводить, що для розрахунку виграшної стратегії можна використовувати невелику кількість параметрів, на відміну від Program1 та Program3, а також відображає, що процедура перерахунку стратегії на кожному етапі гри і реалізація її в межах тільки наступного раунду набагато краща розглянутих при порівнянні алгоритмів.

Висновки

В роботі був розглянутий та описаний алгоритм DeepStack, який був запропонований в [4] та підібрана структура нейронної мережі, за допомогою якої була сформована найменш вразлива стратегія. За результатами експерименту можна сказати, що оптимальною структурою для нейронної мережі буде 4 прихованих шари з функцією активації SoftPlus, оскільки значення вразливість для стратегії, розрахованої за допомогою даної мережі, має найкращий результат.

Надійшла до редакції 23 червня 2017 р.

Література

[1] G. Vohl, B. Bošković and J. Brest, "A Rembrant Poker Bot Program," *Elektrotehniški vestnik*, vol. 79, no. 1-2, pp. 13-18, 2012. **URL:** http://ev.fe.uni-lj.si/1-2-2012/Vohl.pdf

[2] G. Nicolai and R. J. Hilderman, "No-limit texas hold'em poker agents created with evolutionary neural networks," in *Proceedings of the 5th international conference on Computational Intelligence and Games*, Milano, Italy, 2009. **ISBN:** 978-1-4244-4814-2

[3] P. McCurley, "An Artificial Intelligence Agent for Texas Hold'em Poker," 05 08 2009. [Online]. Available: http://poker-ai.org/archive/pokerai.org/public/aith.pdf.

[4] M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson and M. Bowling, "DeepStack: Expert-Level Artificial Intelligence in No-Limit Poker," *Science*, vol. 356, no. 6337, pp. 508-513, 05 May 2017. **DOI:** 10.1126/science.aam6960

[5] M. Zinkevich, M. Johanson, M. Bowling and C. Piccione, "Regret Minimization in Games with Incomplete Information," in *Advances in Neural Information Processing Systems 20*, Vancouver, 2007.

[6] N. Burch, M. Johanson and M. Bowling, "Solving Imperfect Information Games Using Decomposition," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, Quebec, 2014. **URL:** https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8407

[7] F. Southey, M. Bowling, B. Larson, C. Piccione, N. Burch, D. Billings and C. Rayner, "Bayes' Bluff: Opponent Modelling in Poker," in *Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence*, Edinburgh, Scotland, UK, 2005.

URL: https://dslpitt.org/papers/05/p550-southey.pdf

[8] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference for Learning Representations*, San Diego, 2015. arXiv: 1412.6980v9

[9] P. J. Huber, "Robust Estimation of a Location Parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73-101, 1964.

URL: https://projecteuclid.org/download/pdf_1/euclid.aoms/1177703732

[10] G. Walker, "Poker Winrates," 05 2017. [Online]. Available: http://www.thepokerbank.com/strategy/other/winrate.

[11] M. Bowling, N. Burch, M. Johanson and O. Tammelin, "Heads-up limit hold'em poker is solved," *Science*, vol. 347, no. 6218, pp. 145-149, 09 January 2015. **DOI:** 10.1126/science.1259433

[12] "DeepStack for Leduc Hold'em," 2017. [Online]. Available: https://github.com/lifrordi/DeepStack-Leduc.

[13] X. Glorot, A. Bordes and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *Fourteenth International Conference on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, USA, 2011.

URL: http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf

УДК 004.89

Оптимизация нейронной сети алгоритма DeepStack для игры в Leduc Hold'em

Дорогой Я. Ю., к.т.н., доц., ORCID 0000-0003-3848-9852
e-mail argusyk@gmail.com
Цуркан В. В., к.т.н., ORCID 0000-0003-1352-042X
e-mail v.v.tsurkan@gmail.com
Лисовой В. Ю., ORCID 0000-0003-1694-8130
e-mail s.vlad96@gmail.com
Национальный технический университет Украины
"Киевский политехнический институт имени Игоря Сикорского" kpi.ua
Киев, Украина

***Реферат*—В статье рассмотрен вопрос реализации нейронной сети и подбора ее структуры, которая используется в алгоритме DeepStack. Приведено подробное описание алгоритма и принципа его работы. Рассмотренный алгоритм используется для принятия решения во время игры в покер. Покер представлен как игра с неполной информацией. Расчет стратегии происходит на базе двух параметров - контрфактических значений оппонента и диапазона игрока. Предложенная нейронная сеть используется для расчета стратегии, а именно контрфактических значений оппонента. В качестве нейронной сети была выбрана сеть прямого распространения. В качестве данных для обучения использовался набор решеных покерных ситуаций, который включает в себя различные величины ставок и комбинации рук. Рассмотрены несколько структур сетей и выбрана оптимальная. Критерием выбора служит оценка уязвимости стратегии.**

Библ. 13, рис. 9.

Ключевые слова — нейронная сеть; покер; стратегия; контрфактические значения; дерево предсказания.

UDC 004.89

Neural network optimtzation of algorithm DeepStack for playing in Leduc Hold'em

Ya. Yu Dorohyi, PhD, Assoc.Prof., ORCID 0000-0003-3848-9852
e-mail argusyk@gmail.com
V. V. Tsurkan, PhD, ORCID 0000-0003-1352-042X
e-mail v.v.tsurkan@gmail.com
V. Yu. Lisovyi, ORCID 0000-0003-1694-8130
e-mail s.vlad96@gmail.com
National technical university of Ukraine "Igor Sikorsky Kyiv polytechnic institute" kpi.ua
Kyiv, Ukraine

***Abstract*—Artificial intelligence is used to automatically control cars, underwater vehicles, aircraft, rockets, robots, text and speech recognition, medical diagnostics and so on. Also in recent years, artificial intelligence is often used in games - in computer games to find ways in two-dimensional or three-dimensional space, simulating the behavior of units, and in board games and card games. Since in most card games players can not accurately develop a winning strategy because the opponent always operates on certain information is unknown to the player, with the evolution of artificial intelligence the question arose whether such a system is capable of playing at the same level as a person and whether it can win a person.**

The article considers the implementation of a neural network and the selection of its structure, which is used in the DeepStack algorithm. Its detailed description and principle of operation are given. This algorithm is used to make decisions during poker. Poker is represented as a game with incomplete information. The calculation of the strategy is based on two

parameters - the counterfactual values of the opponent and the range of the player. The player's range is the probability distribution of the player's possible hands, taking into account the achievement of some public state. Counterfactual values - the probability of the opponent receiving such a hand, which in any case will be higher than the player's hand. These parameters have been described in detail using the formulas from the game theory.

The proposed neural network is used to calculate the strategy, namely counterfactual values of the opponent. As a neural network, a feedforward network was chosen. The input values for the network are the bet size as a part of the player's total bank and the encoded ranges of players that depend on the open cards. The network output is a vector of counterfactual values for each player and hand, which are then interpreted as a bet size. As the data for training, a set of solved poker situations was used, which included various bet values and hand combinations. The proposed neural network is used to calculate the strategy, namely counterfactual values of the opponent. The dataset consisted of 1 000 000 solved poker situations. Several network structures are considered: a network with 2 hidden layers, an activation function of Tanh, a network with 2 hidden layers, an activation function of ReLU, a network with 2 hidden layers, an activation function of SoftPlus, a network with 4 hidden layers, a Tanh activation function, a network with 4 hidden layers, a ReLU activation function, Network with 4 hidden layers SoftPlus activation function. From these structures, the optimal one was chosen. The criterion of choice is the exploitability of the strategy, which shows the number of chips that a player loses by playing a certain strategy. Of the considered structures, the optimal network has 4 hidden layers with the SoftPlus activation function.

Ref. 13, fig. 9.

Key words — neural network; poker; strategy; counterfactual values; lookahead tree.